

```

1      ; *****
2      ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0
3      ; -----
4      ; Last Update: 26/03/2017
5      ; -----
6      ; Beginning: 04/01/2016
7      ; -----
8      ; Assembler: NASM version 2.11 (trdos386.s)
9      ; -----
10     ; Turkish Rational DOS
11     ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
12     ;
13     ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
14     ; unix386.s (03/01/2016)
15     ;
16     ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
17     ; TRDOS2.ASM (09/11/2011)
18     ;
19     ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
20     ; *****
21
22     KLOAD equ 10000h ; Kernel loading address
23     ; NOTE: Retro UNIX 8086 v1 /boot code loads kernel at 1000h:0000h
24     KCODE equ 08h    ; Code segment descriptor (ring 0)
25     KDATA equ 10h    ; Data segment descriptor (ring 0)
26     ; 19/03/2015
27     UCODE equ 1Bh ; 18h + 3h (ring 3)
28     UDATA equ 23h ; 20h + 3h (ring 3)
29     ; 24/03/2015
30     TSS equ 28h     ; Task state segment descriptor (ring 0)
31     ; 19/03/2015
32     CORE equ 400000h ; Start of USER's virtual/linear address space
33     ; (at the end of the 1st 4MB)
34     ECORE equ 0FFC00000h ; End of USER's virtual address space (4GB - 4MB)
35     ; ULIMIT = (ECORE/4096) - 1 = 0FFBFFh (in GDT)
36
37     ; 27/12/2013
38     ;KEND equ KLOAD + 65536 ; (28/12/2013) (end of kernel space)
39     ; 04/07/2016
40     KEND equ KERNELFSIZE + KLOAD
41
42
43
44     ; IBM PC/AT BIOS ----- 10/06/85 (postequ.inc)
45     ;----- CMOS TABLE LOCATION ADDRESS'S -----
46     CMOS_SECONDS EQU 00H    ; SECONDS (BCD)
47     CMOS_SEC_ALARM EQU 01H  ; SECONDS ALARM (BCD)
48     CMOS_MINUTES EQU 02H    ; MINUTES (BCD)
49     CMOS_MIN_ALARM EQU 03H  ; MINUTES ALARM (BCD)
50     CMOS_HOURS EQU 04H      ; HOURS (BCD)
51     CMOS_HR_ALARM EQU 005H  ; HOURS ALARM (BCD)
52     CMOS_DAY_WEEK EQU 06H    ; DAY OF THE WEEK (BCD)
53     CMOS_DAY_MONTH EQU 07H  ; DAY OF THE MONTH (BCD)
54     CMOS_MONTH EQU 08H      ; MONTH (BCD)
55     CMOS_YEAR EQU 09H       ; YEAR (TWO DIGITS) (BCD)
56     CMOS_CENTURY EQU 32H     ; DATE CENTURY BYTE (BCD)
57     CMOS_REG_A EQU 0AH      ; STATUS REGISTER A
58     CMOS_REG_B EQU 00BH     ; STATUS REGISTER B ALARM
59     CMOS_REG_C EQU 00CH     ; STATUS REGISTER C FLAGS
60     CMOS_REG_D EQU 0DH      ; STATUS REGISTER D BATTERY
61     CMOS_SHUT_DOWN EQU 0FH  ; SHUTDOWN STATUS COMMAND BYTE
62     ;-----
63     ; CMOS EQUATES FOR THIS SYSTEM ;
64     ;-----
65     CMOS_PORT EQU 070H      ; I/O ADDRESS OF CMOS ADDRESS PORT
66     CMOS_DATA EQU 071H      ; I/O ADDRESS OF CMOS DATA PORT
67     NMI EQU 10000000B       ; DISABLE NMI INTERRUPTS MASK -
68     ; HIGH BIT OF CMOS LOCATION ADDRESS
69
70     ; Memory Allocation Table Address
71     ; 05/11/2014
72     ; 31/10/2014
73     MEM_ALLOC_TBL equ 100000h ; Memory Allocation Table at the end of
74     ; the 1st 1 MB memory space.
75     ; (This address must be aligned
76     ; on 128 KB boundary, if it will be
77     ; changed later.)
78     ; ((lower 17 bits of 32 bit M.A.T.
79     ; address must be ZERO)).
80     ; (((Reason: 32 bit allocation
81     ; instructions, dword steps)))
82     ; (((byte >> 12 --> page >> 5)))
83
84     ;04/11/2014
85     PDE_A_PRESENT equ 1      ; Present flag for PDE
86     PDE_A_WRITE equ 2        ; Writable (write permission) flag
87     PDE_A_USER equ 4          ; User (non-system/kernel) page flag
88     ;
89     PTE_A_PRESENT equ 1      ; Present flag for PTE (bit 0)
90     PTE_A_WRITE equ 2        ; Writable (write permission) flag (bit 1)
91     PTE_A_USER equ 4          ; User (non-system/kernel) page flag (bit 2)
92     PTE_A_ACCESS equ 32      ; Accessed flag (bit 5) ; 09/03/2015
93
94     ; 17/02/2015 (unix386.s)
95     ; 10/12/2014 - 30/12/2014 (0B000h -> 9000h) (dsectrm2.s)
96     DPT_SEGM equ 09000h ; FDPT segment (EDD v1.1, EDD v3)
97     ;
98     HD0_DPT equ 0            ; Disk parameter table address for hd0
99     HD1_DPT equ 32           ; Disk parameter table address for hd1
100    HD2_DPT equ 64           ; Disk parameter table address for hd2
101    HD3_DPT equ 96           ; Disk parameter table address for hd3
102

```

```

103 ; FDPT (Phoenix, Enhanced Disk Drive Specification v1.1, v3.0)
104 ; (HDPT: Programmer's Guide to the AMIBIOS, 1993)
105 ;
106 FDPT_CYLS equ 0 ; 1 word, number of cylinders
107 FDPT_HDS equ 2 ; 1 byte, number of heads
108 FDPT_TT equ 3 ; 1 byte, A0h = translated FDPT with logical values
109 ; otherwise it is standard FDPT with physical values
110 FDPT_PCOMP equ 5 ; 1 word, starting write precompensation cylinder
111 ; (obsolete for IDE/ATA drives)
112 FDPT_CB equ 8 ; 1 byte, drive control byte
113 ; Bits 7-6 : Enable or disable retries (00h = enable)
114 ; Bit 5 : 1 = Defect map is located at last cyl. + 1
115 ; Bit 4 : Reserved. Always 0
116 ; Bit 3 : Set to 1 if more than 8 heads
117 ; Bit 2-0 : Reserved. Always 0
118 FDPT_LZ equ 12 ; 1 word, landing zone (obsolete for IDE/ATA drives)
119 FDPT_SPT equ 14 ; 1 byte, sectors per track
120
121 ; Floppy Drive Parameters Table (Programmer's Guide to the AMIBIOS, 1993)
122 ; (11 bytes long) will be used by diskette handler/bios
123 ; which is derived from IBM PC-AT BIOS (DISKETTE.ASM, 21/04/1986).
124
125 ; 01/02/2016
126 Logical_DOSDisks equ 90000h + 100h ; 26*256 = 6656 bytes
127 Directory_Buffer equ 80000h ; max = 64K Bytes
128 FAT_Buffer equ 91C00h ; 1536 bytes (3 sectors)
129 ; 15/02/2016
130 Cluster_Buffer equ 70000h ; max = 64K Bytes ; buffer for file read & write
131 ; 11/04/2016
132 Env_Page: equ 93000h ; 512 bytes (4096 bytes)
133 Env_Page_Size equ 512 ; (4096 bytes)
134 ; 30/07/2016
135 Video_Pg_Backup equ 98000h ; Mode 3h, video page backup (32K, 8 pages)
136
137 [BITS 16] ; We need 16-bit instructions for Real mode
138
139 [ORG 0]
140 ; 12/11/2014
141 ; Save boot drive number (that is default root drive)
142 00000000 8816[525C] mov [boot_drv], dl ; physical drv number
143
144 ; Determine installed memory
145 ; 31/10/2014
146 ;
147 00000004 B801E8 mov ax, 0E801h ; Get memory size
148 00000007 CD15 int 15h ; for large configurations
149 00000009 7308 jnc short chk_ms
150 0000000B B488 mov ah, 88h ; Get extended memory size
151 0000000D CD15 int 15h
152 ;
153 ;mov al, 17h ; Extended memory (1K blocks) low byte
154 ;out 70h, al ; select CMOS register
155 ;in al, 71h ; read data (1 byte)
156 ;mov cl, al
157 ;mov al, 18h ; Extended memory (1K blocks) high byte
158 ;out 70h, al ; select CMOS register
159 ;in al, 71h ; read data (1 byte)
160 ;mov ch, al
161 ;
162 0000000F 89C1 mov cx, ax
163 00000011 31D2 xor dx, dx
164
165 00000013 890E[4E5C] chk_ms: mov [mem_lm_1k], cx
166 00000017 8916[505C] mov [mem_l6m_64k], dx
167 ; 05/11/2014
168 ;and dx, dx
169 ;jz short L2
170 0000001B 81F90004 cmp cx, 1024
171 0000001F 7351 jnb short L0
172 ; insufficient memory_error
173 ; Minimum 2 MB memory is needed...
174 ; 05/11/2014
175 ; (real mode error printing)
176 00000021 FB sti
177 00000022 BE[3600] mov si, msg_out_of_memory
178 00000025 BB0700 mov bx, 7
179 00000028 B40E mov ah, 0Eh ; write tty
180
181 0000002A AC oom_1: lodsb
182 0000002B 08C0 or al, al
183 0000002D 7404 jz short oom_2
184 0000002F CD10 int 10h
185 00000031 EBF7 jmp short oom_1
186
187 00000033 F4 oom_2: hlt
188 00000034 EBF7 jmp short oom_2
189
190 ; 20/02/2017
191 ; 05/11/2014
192 msg_out_of_memory:
193 00000036 070D0A db 07h, 0Dh, 0Ah
194 00000039 496E737566666696369- db 'Insufficient memory !'
194 00000042 656E74206D656D6F72-
194 0000004B 792021
195 0000004E 0D0A db 0Dh, 0Ah
196
197 00000050 284D696E696D756D20- _int13h_48h_buffer: ; 07/07/2016
197 00000059 324D42206D656D6F72- db '(Minimum 2MB memory is needed.)'
197 00000062 79206973206E656564-
197 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[555C] <1> inc byte [hdc] ; count of hard disks (EDD present)
42 0000008A 8816[545C] <1> mov [last_drv], dl ; last hard disk number
43 0000008E BB[D85B] <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[565C] <1> mov si, fd0_type
61 <1> L3:
62 <1> ; 14/01/2015
63 0000009F 8816[535C] <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[565C] <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[555C]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[535C] <1> mov [drv], dl
94 000000CA 8816[545C] <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[555C] <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     <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 ; hdl 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[535C] <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[535C] <1>      inc byte [drv]
148 00000139 BB[D85B] <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[555C] <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[535C] <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[535C] <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[585C] <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[565C] <1>      sub bx, hd0_type - 2 ; 15/01/2015
198 00000195 81C3[A25C] <1>      add bx, drv.status
199 00000199 8807 <1>      mov [bx], al
200     <1>      ; 04/02/2016 (eax -> ax)

```

```

201 0000019B 8B4410      <1>      mov     ax, [si+16]
202 0000019E 854412      <1>      test    ax, [si+18]
203 000001A1 7412       <1>      jz      short L10_A0h
204                               <1>      ; 'CHS only' disks on EDD system
205                               <1>      ; are reported with ZERO disk size
206 000001A3 81EB[A25C]   <1>      sub     bx, drv.status
207 000001A7 C1E302     <1>      shl     bx, 2
208 000001AA 81C3[865C]   <1>      add     bx, drv.size ; disk size (in sectors)
209 000001AE 8907       <1>      mov     [bx], ax
210 000001B0 8B4412      <1>      mov     ax, [si+18]
211 000001B3 8907       <1>      mov     [bx], ax
212                               <1>
213                               <1> L10_A0h: ; Jump here to fix a ZERO (LBA) disk size problem
214                               <1>      ; for CHS disks (28/02/2015)
215                               <1>      ; 30/12/2014
216 000001B5 BF0000     <1>      mov     di, HD0_DPT
217 000001B8 88D0       <1>      mov     al, dl
218 000001BA 83E003     <1>      and     ax, 3
219 000001BD C0E005     <1>      shl     al, 5 ; *32
220 000001C0 01C7       <1>      add     di, ax
221 000001C2 B80090     <1>      mov     ax, DPT_SEGM
222 000001C5 8EC0       <1>      mov     es, ax
223                               <1>      ;
224 000001C7 88E8       <1>      mov     al, ch ; max. cylinder number (bits 0-7)
225 000001C9 88CC       <1>      mov     ah, cl
226 000001CB C0EC06     <1>      shr     ah, 6 ; max. cylinder number (bits 8-9)
227 000001CE 40         <1>      inc     ax ; logical cylinders (limit 1024)
228 000001CF AB         <1>      stosw
229 000001D0 88F0       <1>      mov     al, dh ; max. head number
230 000001D2 FEC0       <1>      inc     al
231 000001D4 AA         <1>      stosb ; logical heads (limits 256)
232 000001D5 B0A0       <1>      mov     al, 0A0h ; Indicates translated table
233 000001D7 AA         <1>      stosb
234 000001D8 8A440C     <1>      mov     al, [si+12]
235 000001DB AA         <1>      stosb ; physical sectors per track
236 000001DC 31C0       <1>      xor     ax, ax
237                               <1>      ;dec     ax ; 02/01/2015
238 000001DE AB         <1>      stosw ; precompensation (obsolete)
239                               <1>      ;xor     al, al ; 02/01/2015
240 000001DF AA         <1>      stosb ; reserved
241 000001E0 B008       <1>      mov     al, 8 ; drive control byte
242                               <1>      ; (do not disable retries,
243                               <1>      ; more than 8 heads)
244                               <1>      stosb
245 000001E3 8B4404     <1>      mov     ax, [si+4]
246 000001E6 AB         <1>      stosw ; physical number of cylinders
247                               <1>      ;push    ax ; 02/01/2015
248 000001E7 8A4408     <1>      mov     al, [si+8]
249 000001EA AA         <1>      stosb ; physical num. of heads (limit 16)
250 000001EB 29C0       <1>      sub     ax, ax
251                               <1>      ;pop     ax ; 02/01/2015
252 000001ED AB         <1>      stosw ; landing zone (obsolete)
253 000001EE 88C8       <1>      mov     al, cl ; logical sectors per track (limit 63)
254 000001F0 243F       <1>      and     al, 3Fh
255 000001F2 AA         <1>      stosb
256                               <1>      ;sub     al, al ; checksum
257                               <1>      ;stosb
258                               <1>      ;
259 000001F3 83C61A     <1>      add     si, 26 ; (BIOS) DPTE address pointer
260 000001F6 AD         <1>      lodsw
261 000001F7 50         <1>      push    ax ; (BIOS) DPTE offset
262 000001F8 AD         <1>      lodsw
263 000001F9 50         <1>      push    ax ; (BIOS) DPTE segment
264                               <1>      ;
265                               <1>      ; checksum calculation
266 000001FA 89FE       <1>      mov     si, di
267 000001FC 06         <1>      push    es
268 000001FD 1F         <1>      pop     ds
269                               <1>      ;mov     cx, 16
270 000001FE B90F00     <1>      mov     cx, 15
271 00000201 29CE       <1>      sub     si, cx
272 00000203 30E4       <1>      xor     ah, ah
273                               <1>      ;del     cl
274                               <1> L11:
275 00000205 AC         <1>      lodsb
276 00000206 00C4       <1>      add     ah, al
277 00000208 E2FB       <1>      loop    L11
278                               <1>      ;
279 0000020A 88E0       <1>      mov     al, ah
280 0000020C F6D8       <1>      neg     al ; -x+x = 0
281 0000020E AA         <1>      stosb ; put checksum in byte 15 of the tbl
282                               <1>      ;
283 0000020F 1F         <1>      pop     ds ; (BIOS) DPTE segment
284 00000210 5E         <1>      pop     si ; (BIOS) DPTE offset
285                               <1>      ;
286                               <1>      ; 23/02/2015
287 00000211 57         <1>      push    di
288                               <1>      ; ES:DI points to DPTE (FDPTE) location
289                               <1>      ;mov     cx, 8
290 00000212 B108       <1>      mov     cl, 8
291 00000214 F3A5       <1>      rep     movsw
292                               <1>      ;
293                               <1>      ; 23/02/2015
294                               <1>      ; (P)ATA drive and LBA validation
295                               <1>      ; (invalidating SATA drives and setting
296                               <1>      ; CHS type I/O for old type fixed disks)
297 00000216 5B         <1>      pop     bx
298 00000217 8CC8       <1>      mov     ax, cs
299 00000219 8ED8       <1>      mov     ds, ax
300 0000021B 268B07     <1>      mov     ax, [es:bx]
301 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[585C]00 <1>      mov     byte [bx+hd0_type], 0 ; not a valid disk drive !
311 00000234 808F[A45C]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[A45C]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[545C] <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[535C] <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[A25C] <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[A25C] <1>      sub     bx, drv.status
363 0000028A D0E3      <1>      shl     bl, 1
364 0000028C 81C3[5C5C] <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[5C5C] <1>      sub     bx, drv.cylinders
369 00000298 81C3[6A5C] <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[6A5C] <1>      sub     bx, drv.heads
375 000002A7 81C3[785C] <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[785C] <1>      sub     bx, drv.spt
380 000002B6 D1E3      <1>      shl     bx, 1
381 000002B8 81C3[865C] <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
395                      <1>
396                      <1> ; 10/11/2014
397                      <1> cli ; Disable interrupts (clear interrupt flag)
398                      <1> ; Reset Interrupt MASK Registers (Master&Slave)
399                      <1> ;mov al, 0FFh ; mask off all interrupts
400                      <1> ;out 21h, al ; on master PIC (8259)
401                      <1> ;jmp $+2 ; (delay)
402                      <1> ;out 0A1h, al ; on slave PIC (8259)
403                      <1> ;
404                      <1> ; Disable NMI

```

```

213 000002CB B080      mov     al, 80h
214 000002CD E670      out      70h, al          ; set bit 7 to 1 for disabling NMI
215                      ;23/02/2015
216                      ;nop
217                      ;in      al, 71h          ; read in 71h just after writing out to 70h
218                      ; for preventing unknown state (!?)
219                      ;
220                      ; 20/08/2014
221                      ; Moving the kernel 64 KB back (to physical address 0)
222                      ; DS = CS = 1000h
223                      ; 05/11/2014
224 000002CF 31C0      xor      ax, ax
225 000002D1 8EC0      mov      es, ax ; ES = 0
226                      ;
227                      ; 04/07/2016 - TRDOS 386 (64K - 128K kernel)
228 000002D3 31F6      xor      si, si
229 000002D5 31FF      xor      di, di
230 000002D7 B90040     mov      cx, 16384
231 000002DA F366A5     rep      movsd
232                      ;
233 000002DD 06         push     es ; 0
234 000002DE 68[E202]   push     L17
235 000002E1 CB         retf
236                      L17:
237 000002E2 B90010     mov      cx, 1000h
238 000002E5 8EC1      mov      es, cx ; 1000h
239 000002E7 01C9      add      cx, cx
240 000002E9 8ED9      mov      ds, cx ; 2000h
241 000002EB 29F6      sub      si, si
242 000002ED 29FF      sub      di, di
243 000002EF B90040     mov      cx, 16384
244 000002F2 F366A5     rep      movsd
245                      ;
246                      ; Turn off the floppy drive motor
247 000002F5 BAF203     mov      dx, 3F2h
248 000002F8 EE        out      dx, al ; 0 ; 31/12/2013
249                      ;
250                      ; Enable access to memory above one megabyte
251                      L18:
252 000002F9 E464      in       al, 64h
253 000002FB A802      test     al, 2
254 000002FD 75FA      jnz      short L18
255 000002FF B0D1      mov      al, 0D1h ; Write output port
256 00003301 E664      out      64h, al
257                      L19:
258 00003303 E464      in       al, 64h
259 00003305 A802      test     al, 2
260 00003307 75FA      jnz      short L19
261 00003309 B0DF      mov      al, 0DFh ; Enable A20 line
262 0000330B E660      out      60h, al
263                      ;L20:
264                      ;
265                      ; Load global descriptor table register
266                      ;
267                      ;mov      ax, cs
268                      ;mov      ds, ax
269                      ;
270 0000330D 2E0F0116[C05B] lgdt      [cs:gdt]
271                      ;
272 00003313 0F20C0     mov      eax, cr0
273                      ; or      eax, 1
274 00003316 40         inc      ax
275 00003317 0F22C0     mov      cr0, eax
276                      ;
277                      ; Jump to 32 bit code
278                      ;
279 0000331A 66         db 66h ; Prefix for 32-bit
280 0000331B EA         db 0EAh ; Opcode for far jump
281 0000331C [22030000] dd StartPM ; Offset to start, 32-bit
282                      ; (1000h:StartPM = StartPM + 10000h)
283 00003320 0800      dw KCODE ; This is the selector for CODE32_DESCRIPTOR,
284                      ; assuming that StartPM resides in code32
285                      ;
286                      ; 20/02/2017
287                      ;
288                      ;
289                      [BITS 32]
290                      ;
291                      StartPM:
292                      ; Kernel Base Address = 0 ; 30/12/2013
293 00003322 66B81000   mov      ax, KDATA ; Save data segment identifier
294 00003326 8ED8      mov      ds, ax ; Move a valid data segment into DS register
295 00003328 8EC0      mov      es, ax ; Move data segment into ES register
296 0000332A 8EE0      mov      fs, ax ; Move data segment into FS register
297 0000332C 8EE8      mov      gs, ax ; Move data segment into GS register
298 0000332E 8ED0      mov      ss, ax ; Move data segment into SS register
299 00003330 BC00000900 mov      esp, 90000h ; Move the stack pointer to 090000h
300                      ;
301                      clear_bss: ; Clear uninitialized data area
302                      ; 11/03/2015
303 00003335 31C0      xor      eax, eax ; 0
304 00003337 B9EB090000   mov      ecx, (bss_end - bss_start)/4
305                      ;shr      ecx, 2 ; bss section is already aligned for double words
306 0000333C BF[36350100] mov      edi, bss_start
307 00003341 F3AB      rep      stosd
308                      ;
309                      memory_init:
310                      ; Initialize memory allocation table and page tables
311                      ; 16/11/2014
312                      ; 15/11/2014
313                      ; 07/11/2014
314                      ; 06/11/2014

```

```

315 ; 05/11/2014
316 ; 04/11/2014
317 ; 31/10/2014 (Retro UNIX 386 v1 - Beginning)
318 ;
319 ; xor    eax, eax
320 ; xor    ecx, ecx
321 00000343 B108      mov    cl, 8
322 00000345 BF00001000 mov    edi, MEM_ALLOC_TBL
323 0000034A F3AB      rep    stosd      ; clear Memory Allocation Table
324                      ; for the first 1 MB memory
325 ;
326 0000034C 668B0D[4E5C0000] mov    cx, [mem_1m_1k]      ; Number of contiguous KB between
327                      ; 1 and 16 MB, max. 3C00h = 15 MB.
328 00000353 66C1E902      shr    cx, 2      ; convert 1 KB count to 4 KB count
329 00000357 890D[28380100] mov    [free_pages], ecx
330 0000035D 668B15[505C0000] mov    dx, [mem_16m_64k] ; Number of contiguous 64 KB blocks
331                      ; between 16 MB and 4 GB.
332 00000364 6609D2      or     dx, dx
333 00000367 7413      jz     short mi_0
334 ;
335 00000369 6689D0      mov    ax, dx
336 0000036C C1E004      shl    eax, 4      ; 64 KB -> 4 KB (page count)
337 0000036F 0105[28380100] add    [free_pages], eax
338 00000375 0500100000 add    eax, 4096     ; 16 MB = 4096 pages
339 0000037A EB07      jmp    short mi_1
340
341 0000037C 6689C8      mov    ax, cx
342 0000037F 66050001 add    ax, 256       ; add 256 pages for the first 1 MB
343
344 00000383 A3[24380100] mi_1: mov    [memory_size], eax ; Total available memory in pages
345                      ; 1 alloc. tbl. bit = 1 memory page
346                      ; 32 allocation bits = 32 mem. pages
347 ;
348 00000388 05FF7F0000 add    eax, 32767    ; 32768 memory pages per 1 M.A.T. page
349 0000038D C1E80F      shr    eax, 15      ; ((32768 * x) + y) pages (y < 32768)
350                      ; --> x + 1 M.A.T. pages, if y > 0
351                      ; --> x M.A.T. pages, if y = 0
352 00000390 66A3[38380100] mov    [mat_size], ax ; Memory Alloc. Table Size in pages
353 00000396 C1E00C      shl    eax, 12      ; 1 M.A.T. page = 4096 bytes
354                      ; Max. 32 M.A.T. pages (4 GB memory)
355 00000399 89C3      mov    ebx, eax      ; M.A.T. size in bytes
356                      ; Set/Calculate Kernel's Page Directory Address
357 0000039B 81C300001000 add    ebx, MEM_ALLOC_TBL
358 000003A1 891D[20380100] mov    [k_page_dir], ebx ; Kernel's Page Directory address
359                      ; just after the last M.A.T. page
360 ;
361 000003A7 83E804      sub    eax, 4      ; convert M.A.T. size to offset value
362 000003AA A3[30380100] mov    [last_page], eax ; last page offset in the M.A.T.
363                      ; (allocation status search must be
364                      ; stopped after here)
365 000003AF 31C0      xor    eax, eax
366 000003B1 48      dec    eax      ; FFFFFFFFh (set all bits to 1)
367 000003B2 6651      push   cx
368 000003B4 C1E905      shr    ecx, 5      ; convert 1 - 16 MB page count to
369                      ; count of 32 allocation bits
370 000003B7 F3AB      rep    stosd
371 000003B9 6659      pop     cx
372 000003BB 40      inc    eax      ; 0
373 000003BC 80E11F      and    cl, 31      ; remain bits
374 000003BF 7412      jz     short mi_4
375 000003C1 8907      mov    [edi], eax ; reset
376
377 000003C3 0FAB07      bts    [edi], eax ; 06/11/2014
378 000003C6 FEC9      dec    cl
379 000003C8 7404      jz     short mi_3
380 000003CA FEC0      inc    al
381 000003CC EBF5      jmp    short mi_2
382
383 000003CE 28C0      sub    al, al      ; 0
384 000003D0 83C704 add    edi, 4      ; 15/11/2014
385
386 000003D3 6609D2      or     dx, dx      ; check 16M to 4G memory space
387 000003D6 7421      jz     short mi_6 ; max. 16 MB memory, no more...
388 ;
389 000003D8 B900021000 mov    ecx, MEM_ALLOC_TBL + 512 ; End of first 16 MB memory
390 ;
391 000003DD 29F9      sub    ecx, edi ; displacement (to end of 16 MB)
392 000003DF 7406      jz     short mi_5 ; jump if EDI points to
393                      ; end of first 16 MB
394 000003E1 D1E9      shr    ecx, 1      ; convert to dword count
395 000003E3 D1E9      shr    ecx, 1      ; (shift 2 bits right)
396 000003E5 F3AB      rep    stosd      ; reset all bits for reserved pages
397                      ; (memory hole under 16 MB)
398
399 000003E7 6689D1      mov    cx, dx      ; count of 64 KB memory blocks
400 000003EA D1E9      shr    ecx, 1      ; 1 alloc. dword per 128 KB memory
401 000003EC 9C      pushf ; 16/11/2014
402 000003ED 48      dec    eax      ; FFFFFFFFh (set all bits to 1)
403 000003EE F3AB      rep    stosd
404 000003F0 40      inc    eax      ; 0
405 000003F1 9D      popf ; 16/11/2014
406 000003F2 7305      jnc    short mi_6
407 000003F4 6648      dec    ax      ; eax = 0000FFFFh
408 000003F6 AB      stosd
409 000003F7 6640      inc    ax      ; 0
410
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
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[38380100] 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[28380100] sub     [free_pages], ecx ; 07/11/2014
436
437 00000420 0FB307      mi_8:  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
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[24380100] 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[28380100] sub     [free_pages], ecx ; 07/11/2014
459                          ;
460 00000442 8B35[20380100] mov     esi, [k_page_dir] ; Kernel's Page Directory address
461 00000448 C1EE0C          shr     esi, 12          ; convert to page number
462
463 0000044B 89F0      mi_10: 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[20380100] 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
490 0000046D 0500100000      mi_11: 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
502                          ; Initialize Kernel's Page Tables
503                          ;
504                          ; (EDI points to address of page table 0)
505                          ; eax = 0
506 00000481 8B0D[24380100] 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
510
511 0000048B AB      mi_13: stosd
512 0000048C 0500100000      add     eax, 4096
513 00000491 E2F8          loop   mi_13
514 00000493 6681E2FF03 and     dx, 1023          ; (***)
515 00000498 740B          jz      short mi_14
516 0000049A 66B90004      mov     cx, 1024
517 0000049E 6629D1          sub     cx, dx            ; from dx (<= 1023) to 1024
518 000004A1 31C0          xor     eax, eax

```

```

519 000004A3 F3AB          rep    stosd      ; clear remain (empty) PTEs
520                                ; of the last page table
521
522          mi_14:          ; Initialization of Kernel's Page Tables is OK, here.
523                                ;
524 000004A5 89F8          mov     eax, edi      ; end of the last page table page
525                                ; (beginning of user space pages)
526 000004A7 C1E80F        shr     eax, 15      ; convert to M.A.T. byte offset
527 000004AA 24FC          and     al, 0FCh    ; clear bit 0 and bit 1 for
528                                ; aligning on dword boundary
529
530 000004AC A3[34380100]   mov     [first_page], eax
531 000004B1 A3[2C380100]   mov     [next_page], eax ; The first free page pointer
532                                ; for user programs
533                                ; (Offset in Mem. Alloc. Tbl.)
534
535          ; Linear/FLAT (1 to 1) memory paging for the kernel is OK, here.
536          ;
537
538          ; Enable paging
539          ;
540 000004B6 A1[20380100]   mov     eax, [k_page_dir]
541 000004BB 0F22D8        mov     cr3, eax
542 000004BE 0F20C0        mov     eax, cr0
543 000004C1 0D00000080    or      eax, 80000000h    ; set paging bit (bit 31)
544 000004C6 0F22C0        mov     cr0, eax
545                                ; jmp     KCODE:StartPMP
546
547 000004C9 EA            db 0EAh          ; Opcode for far jump
548 000004CA [D0040000]     dd StartPMP      ; 32 bit offset
549 000004CE 0800        dw KCODE          ; kernel code segment descriptor
550
551
552          StartPMP:
553          ; 06/11//2014
554          ; Clear video page 0
555          ;
556          ; Temporary Code
557          ;
558 000004D0 B9E8030000    mov     ecx, 80*25/2
559 000004D5 BF00800B00    mov     edi, 0B8000h
560          ; 30/01/2016
561          ; xor     eax, eax      ; black background, black fore color
562 000004DA B800070007    mov     eax, 07000700h ; black background, light gray fore color
563 000004DF F3AB          rep     stosd
564
565          ; 19/08/2014
566          ; Kernel Base Address = 0
567          ; It is mapped to (physically) 0 in the page table.
568          ; So, here is exactly 'StartPMP' address.
569
570          ; 29/01/2016 (TRDOS 386 = TRDOS v2.0)
571 000004E1 BE[31070100]   mov     esi, starting_msg
572          ; ; 14/08/2015 (kernel version message will appear
573          ; ; when protected mode and paging is enabled)
574 000004E6 BF00800B00    mov     edi, 0B8000h ; 27/08/2014
575 000004EB B40A          mov     ah, 0Ah ; Black background, light green forecolor
576          ; 20/08/2014
577 000004ED E88F010000    call    printk
578
579          ; 'UNIX v7/x86' source code by Robert Nordier (1999)
580          ; // Set IRQ offsets
581          ;
582          ; Linux (v0.12) source code by Linus Torvalds (1991)
583          ;
584          ; ; ICW1
585 000004F2 B011          mov     al, 11h      ; Initialization sequence
586 000004F4 E620          out     20h, al      ; 8259A-1
587          ; jmp     $+2
588 000004F6 E6A0          out     0A0h, al     ; 8259A-2
589          ; ; ICW2
590 000004F8 B020          mov     al, 20h      ; Start of hardware ints (20h)
591 000004FA E621          out     21h, al      ; for 8259A-1
592          ; jmp     $+2
593 000004FC B028          mov     al, 28h      ; Start of hardware ints (28h)
594 000004FE E6A1          out     0A1h, al     ; for 8259A-2
595          ;
596 00000500 B004          mov     al, 04h      ; ; ICW3
597 00000502 E621          out     21h, al      ; IRQ2 of 8259A-1 (master)
598          ; jmp     $+2
599 00000504 B002          mov     al, 02h      ; is 8259A-2 (slave)
600 00000506 E6A1          out     0A1h, al     ;
601          ; ; ICW4
602 00000508 B001          mov     al, 01h      ;
603 0000050A E621          out     21h, al      ; 8086 mode, normal EOI
604          ; jmp     $+2
605 0000050C E6A1          out     0A1h, al     ; for both chips.
606
607          ; mov     al, 0FFh    ; mask off all interrupts for now
608          ; out     21h, al
609          ; ; jmp     $+2
610          ; out     0A1h, al
611
612          ; 02/04/2015
613          ; 26/03/2015 System call (INT 30h) modification
614          ; DPL = 3 (Interrupt service routine can be called from user mode)
615          ;
616          ; ; Linux (v0.12) source code by Linus Torvalds (1991)
617          ; setup_idt:
618          ;
619          ; ; 16/02/2015
620          ; mov     dword [DISKETTE_INT], fdc_int ; IRQ 6 handler

```

```

621 ; 21/08/2014 (timer_int)
622 0000050E BE[F4030100] mov esi, ilist
623 00000513 8D3D[38350100] lea edi, [idt]
624 ; 26/03/2015
625 00000519 B930000000 mov ecx, 48 ; 48 hardware interrupts (INT 0 to INT 2Fh)
626 ; 02/04/2015
627 0000051E BB00000800 mov ebx, 80000h
628 rp_sidtl:
629 00000523 AD lodsd
630 00000524 89C2 mov edx, eax
631 00000526 66BA008E mov dx, 8E00h
632 0000052A 6689C3 mov bx, ax
633 0000052D 89D8 mov eax, ebx ; /* selector = 0x0008 = cs */
634 ; /* interrupt gate - dpl=0, present */
635 0000052F AB stosd ; selector & offset bits 0-15
636 00000530 89D0 mov eax, edx
637 00000532 AB stosd ; attributes & offset bits 16-23
638 00000533 E2EE loop rp_sidtl
639 ; 15/04/2016
640 ; TRDOS 386 (TRDOS v2.0) /// 32 software interrupts ///
641 ;mov cl, 16 ; 16 software interrupts (INT 30h to INT 3Fh)
642 00000535 B120 mov cl, 32 ; 32 software interrupts (INT 30h to INT 4Fh)
643 rp_sidt2:
644 00000537 AD lodsd
645 00000538 21C0 and eax, eax
646 0000053A 7413 jz short rp_sidt3
647 0000053C 89C2 mov edx, eax
648 0000053E 66BA00EE mov dx, 0EE00h ; P=1b/DPL=11b/01110b
649 00000542 6689C3 mov bx, ax
650 00000545 89D8 mov eax, ebx ; selector & offset bits 0-15
651 00000547 AB stosd
652 00000548 89D0 mov eax, edx
653 0000054A AB stosd
654 0000054B E2EA loop rp_sidt2
655 0000054D EB16 jmp short sidt_OK
656 rp_sidt3:
657 0000054F B8[AA0A0000] mov eax, ignore_int
658 00000554 89C2 mov edx, eax
659 00000556 66BA00EE mov dx, 0EE00h ; P=1b/DPL=11b/01110b
660 0000055A 6689C3 mov bx, ax
661 0000055D 89D8 mov eax, ebx ; selector & offset bits 0-15
662 rp_sidt4:
663 0000055F AB stosd
664 00000560 92 xchg eax, edx
665 00000561 AB stosd
666 00000562 92 xchg edx, eax
667 00000563 E2FA loop rp_sidt4
668 sidt_OK:
669 00000565 0F011D[C65B0000] lidt [idtd]
670 ;
671 ; TSS descriptor setup ; 24/03/2015
672 0000056C B8[B8370100] mov eax, task_state_segment
673 00000571 66A3[BA5B0000] mov [gdt_tss0], ax
674 00000577 C1C010 rol eax, 16
675 0000057A A2[BC5B0000] mov [gdt_tss1], al
676 0000057F 8825[BF5B0000] mov [gdt_tss2], ah
677 00000585 66C705[1E380100]68- mov word [tss.IOPB], tss_end - task_state_segment
677 0000058D 00
678 ;
679 ; IO Map Base address (When this address points
680 ; to end of the TSS, CPU does not use IO port
681 ; permission bit map for RING 3 IO permissions,
682 ; access to any IO ports in ring 3 will be forbidden.)
683 ;
684 ;mov [tss.esp0], esp ; TSS offset 4
685 ;mov word [tss.ss0], KDATA ; TSS offset 8 (SS)
686 0000058E 66B82800 mov ax, TSS ; It is needed when an interrupt
687 ; occurs (or a system call -software INT- is requested)
688 ; while cpu running in ring 3 (in user mode).
689 ; (Kernel stack pointer and segment will be loaded
690 ; from offset 4 and 8 of the TSS, by the CPU.)
691 00000592 0F00D8 ltr ax ; Load task register
692 ;
693 esp0_set0:
694 ; 30/07/2015
695 00000595 8B0D[24380100] mov ecx, [memory_size] ; memory size in pages
696 0000059B C1E10C shl ecx, 12 ; convert page count to byte count
697 0000059E 81F900004000 cmp ecx, CORE ; beginning of user's memory space (400000h)
698 ; (kernel mode virtual address)
699 000005A4 7605 jna short esp0_set1
700 ;
701 ; If available memory > CORE (end of the 1st 4 MB)
702 ; set stack pointer to CORE
703 ;(Because, PDE 0 is reserved for kernel space in user's page directory)
704 ;(PDE 0 points to page table of the 1st 4 MB virtual address space)
705 000005A6 B900004000 mov ecx, CORE
706 esp0_set1:
707 000005AB 89CC mov esp, ecx ; top of kernel stack (**tss.esp0**)
708 esp0_set_ok:
709 ; 30/07/2015 (**tss.esp0**)
710 000005AD 8925[BC370100] mov [tss.esp0], esp
711 000005B3 66C705[C0370100]10- mov word [tss.ss0], KDATA
711 000005BB 00
712 ; 14/08/2015
713 ; 10/11/2014 (Retro UNIX 386 v1 - Erdogan Tan)
714 ;
715 ;cli ; Disable interrupts (for CPU)
716 ; (CPU will not handle hardware interrupts, except NMI!)
717 ;
718 000005BC 30C0 xor al, al ; Enable all hardware interrupts!
719 000005BE E621 out 21h, al ; (IBM PC-AT compatibility)
720 000005C0 EB00 jmp $+2 ; (All conventional PC-AT hardware

```

```

721 000005C2 E6A1      out    0Ah, al      ; interrupts will be in use.)
722                      ; (Even if related hardware component
723                      ; does not exist!)
724                      ; Enable NMI
725 000005C4 B07F      mov     al, 7Fh      ; Clear bit 7 to enable NMI (again)
726 000005C6 E670      out     70h, al
727                      ; 23/02/2015
728 000005C8 90        nop
729 000005C9 E471      in      al, 71h      ; read in 71h just after writing out to 70h
730                      ; for preventing unknown state (!?)
731                      ;
732                      ; Only a NMI can occur here... (Before a 'STI' instruction)
733                      ;
734                      ; 02/09/2014
735 000005CB 6631DB     xor     bx, bx
736 000005CE 66BA0002   mov     dx, 0200h    ; Row 2, column 0 ; 07/03/2015
737 000005D2 E871170000 call    _set_cpos    ; 24/01/2016
738                      ;
739                      ; 06/11/2014
740 000005D7 E8782C0000 call    memory_info
741                      ; 14/08/2015
742                      ;call getch ; 28/02/2015
743 drv_init:
744 000005DC FB        sti     ; Enable Interrupts
745                      ; 06/02/2015
746 000005DD 8B15[585C0000] mov     edx, [hd0_type] ; hd0, hd1, hd2, hd3
747 000005E3 668B1D[565C0000] mov     bx, [fd0_type] ; fd0, fd1
748                      ; 22/02/2015
749 000005EA 6621DB     and     bx, bx
750 000005ED 751C      jnz     short di1
751                      ;
752 000005EF 09D2      or      edx, edx
753 000005F1 752A      jnz     short di2
754                      ;
755 setup_error:
756 000005F3 BE[FA060100] mov     esi, setup_error_msg
757 psem:
758 000005F8 AC        lodsb
759 000005F9 08C0      or      al, al
760                      ;jz     short haltx ; 22/02/2015
761 000005FB 7427      jz      short di3
762 000005FD 56        push    esi
763                      ; 13/05/2016
764 000005FE BB07000000 mov     ebx, 7 ; Black background,
765                      ; light gray forecolor
766                      ; Video page 0 (BH=0)
767 00000603 E8AA160000 call    _write_tty
768 00000608 5E        pop     esi
769 00000609 EBED      jmp     short psem
770
771 dil:
772                      ; supress 'jmp short T6'
773                      ; (activate fdc motor control code)
774 0000060B 66C705[EB060000]90- mov     word [T5], 9090h ; nop
775                      ;
776                      ;mov    ax, int_0Eh ; IRQ 6 handler
777                      ;mov    di, 0Eh*4 ; IRQ 6 vector
778                      ;stosw
779                      ;mov    ax, cs
780                      ;stosw
781                      ;; 16/02/2015
782                      ;;mov    dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
783                      ;
784 00000614 E8AF3B0000 CALL    DSKETTE_SETUP; Initialize Floppy Disks
785                      ;
786 00000619 09D2      or      edx, edx
787 0000061B 7407      jz      short di3
788 di2:
789 0000061D E8EC3B0000 call    DISK_SETUP ; Initialize Fixed Disks
790 00000622 72CF      jc      short setup_error
791 di3:
792 00000624 E8FF2B0000 call    setup_rtc_int; 22/05/2015 (dsctrpm.s)
793                      ;
794 00000629 E862FF0000 call    display_disks ; 07/03/2015 (Temporary)
795 ;haltx:
796                      ; 14/08/2015
797                      ;call    getch ; 22/02/2015
798                      ;sti     ; Enable interrupts (for CPU)
799                      ; 29/01/2016
800                      ; sub    ah, ah ; read time count
801                      ; call    int1Ah
802                      ; mov    edx, ecx ; 18.2 * seconds
803 ;md_info_msg_wait1:
804                      ; 29/01/2016
805                      ; mov    ah, 1
806                      ; call    int16h
807                      ; jz     short md_info_msg_wait2
808                      ; xor    ah, ah ; 0
809                      ; call    int16h
810                      ; jmp     short md_info_msg_ok
811 ;md_info_msg_wait2:
812                      ; sub    ah, ah ; read time count
813                      ; call    int1Ah
814                      ; cmp    edx, ecx ; ; 18.2 * seconds
815                      ; jna     short md_info_msg_wait3
816                      ; xchg    edx, ecx
817 ;md_info_msg_wait3:
818                      ; sub    ecx, edx
819                      ; cmp    ecx, 127 ; 7 seconds (18.2 * 7)
820                      ; jnb     short md_info_msg_wait1
821 ;md_info_msg_ok:

```

```

822 ; 08/09/2016
823 mov     eax, cr0
824 test    al, 10h ; Bit 4, ET (Extension Type)
825 jz      short sysinit
826 ; 27/02/2017
827 inc     byte [fpready]
828 ; 80387 (FPU) is ready
829 fninit ; Initialize Floating-Point Unit
830 sysinit:
831 ; 30/06/2015
832 call    sys_init
833 ;
834 ; jmp     cpu_reset ; 22/02/2015
835 hang:
836 ; 23/02/2015
837 ; sti             ; Enable interrupts
838 hlt
839 ;
840 ; nop
841 ; ; 03/12/2014
842 ; ; 28/08/2014
843 ; mov     ah, 11h
844 ; call    getc
845 ; jz      _c8
846 ;
847 ; 23/02/2015
848 ; 06/02/2015
849 ; 07/09/2014
850 xor     ebx, ebx
851 mov     bl, [ptty] ; active_page
852 mov     esi, ebx
853 shl     si, 1
854 add     esi, ttychr
855 mov     ax, [esi]
856 and     ax, ax
857 ; jz      short _c8
858 jz      short hang
859 mov     word [esi], 0
860 cmp     bl, 3 ; Video page 3
861 ; jb      short _c8
862 jb      short hang
863 ;
864 ; 13/05/2016
865 ; 07/09/2014
866 nxtl:
867 push    bx
868 mov     bx, 0Eh ; Yellow character
869 ; on black background
870 ; bh = 0 (video page 0)
871 ; Retro UNIX 386 v1 - Video Mode 0
872 ; (PC/AT Video Mode 3 - 80x25 Alpha.)
873 push    ax
874 call    _write_tty
875 pop     ax
876 pop     bx
877 cmp     al, 0Dh ; carriage return (enter)
878 ; jne     short _c8
879 jne     short hang
880 mov     al, 0Ah ; next line
881 jmp     short nxtl
882 ;
883 ; _c8:
884 ; ; 25/08/2014
885 ; cli             ; Disable interrupts
886 ; mov     al, [scounter + 1]
887 ; and     al, al
888 ; jnz     hang
889 ; call    rtc_p
890 ; jmp     hang
891 ;
892 ; 27/08/2014
893 ; 20/08/2014
894 printk:
895 ; mov     edi, [scr_row]
896 ;
897 ;
898 lodsb
899 or      al, al
900 jz      short pkr
901 stosw
902 jmp     short pkl
903 pkr:
904 retn
905 ;
906 ; 28/02/2017
907 ; 22/01/2017
908 ; 15/01/2017
909 ; 14/01/2017
910 ; 02/01/2017
911 ; 25/12/2016
912 ; 19/12/2016
913 ; 10/12/2016 (callback)
914 ; 06/06/2016
915 ; 23/05/2016
916 ; 22/05/2016 - TRDOS 386 (TRDOS v2.0) Timer Event Modifications
917 ; 25/07/2015
918 ; 14/05/2015 (multi tasking -time sharing- 'clock', x_timer)
919 ; 17/02/2015
920 ; 06/02/2015 (unix386.s)
921 ; 11/12/2014 - 22/12/2014 (dsectrm2.s)
922 ;
923 ; IBM PC-XT Model 286 Source Code - BIOS2.ASM (06/10/85)

```



```

924 ;
925 ;-- HARDWARE INT 08 H - ( IRQ LEVEL 0 ) -----
926 ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF :
927 ; THE 8254 TIMER. INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR :
928 ; IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND. :
929 ; :
930 ; THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE :
931 ; POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY. :
932 ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40) :
933 ; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE :
934 ; DISKETTE MOTOR(s), AND RESET THE MOTOR RUNNING FLAGS. :
935 ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH :
936 ; INTERRUPT 1CH AT EVERY TIME TICK. THE USER MUST CODE A :
937 ; ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE. :
938 ;-----
939 ;
940
941 timer_int: ; IRQ 0
942 ;int_08h: ; Timer
943 ; 14/10/2015
944 ; Here, we are simulating system call entry (for task switch)
945 ; (If multitasking is enabled,
946 ; 'clock' procedure may jump to 'sysrelease')
947
948 0000068B 1E push ds
949 0000068C 06 push es
950 0000068D 0FA0 push fs
951 0000068F 0FA8 push gs
952
953 00000691 60 pushad ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
954 00000692 66B91000 mov cx, KDATA
955 00000696 8ED9 mov ds, cx
956 00000698 8EC1 mov es, cx
957 0000069A 8EE1 mov fs, cx
958 0000069C 8EE9 mov gs, cx
959
960 0000069E 0F20D9 mov ecx, cr3
961 000006A1 890D[A44F0100] mov [cr3reg], ecx ; save current cr3 register value/content
962
963 ; 14/01/2017
964 000006A7 3B0D[20380100] cmp ecx, [k_page_dir]
965 000006AD 7409 je short T3
966
967 000006AF 8B0D[20380100] mov ecx, [k_page_dir]
968 000006B5 0F22D9 mov cr3, ecx
969
970 T3: ;sti ; INTERRUPTS BACK ON
971 000006B8 66FF05[A0380100] INC word [TIMER_LOW] ; INCREMENT TIME
972 000006BF 7507 JNZ short T4 ; GO TO TEST_DAY
973 000006C1 66FF05[A2380100] INC word [TIMER_HIGH] ; INCREMENT HIGH WORD OF TIME
974 T4: ; TEST_DAY
975 000006C8 66833D[A2380100]18 CMP word [TIMER_HIGH],018H ; TEST FOR COUNT EQUALING 24 HOURS
976 000006D0 7519 JNZ short T5 ; GO TO DISKETTE_CTL
977 000006D2 66813D[A0380100]B0- CMP word [TIMER_LOW],0B0H
977 000006DA 00
978 000006DB 750E JNZ short T5 ; GO TO DISKETTE_CTL
979
980 ;----- TIMER HAS GONE 24 HOURS
981 ;SUB AX,AX
982 ;MOV [TIMER_HIGH],AX
983 ;MOV [TIMER_LOW],AX
984 000006DD 29C0 sub eax, eax
985 000006DF A3[A0380100] mov [TIMER_LH], eax
986 ;
987 000006E4 C605[A4380100]01 MOV byte [TIMER_OFL],1
988
989 ;----- TEST FOR DISKETTE TIME OUT
990
991 T5:
992 ; 23/12/2014
993 000006EB EB1D jmp short T6 ; will be replaced with nop, nop
994 ; (9090h) if a floppy disk
995 ; is detected.
996 ;mov al,[CS:MOTOR_COUNT]
997 000006ED A0[A7380100] mov al, [MOTOR_COUNT]
998 000006F2 FEC8 dec al
999 ;mov [CS:MOTOR_COUNT], al ; DECREMENT DISKETTE MOTOR CONTROL
1000 000006F4 A2[A7380100] mov [MOTOR_COUNT], al
1001 ;mov [ORG_MOTOR_COUNT], al
1002 000006F9 750F JNZ short T6 ; RETURN IF COUNT NOT OUT
1003 000006FB B0F0 mov al,0F0h
1004 ;AND [CS:MOTOR_STATUS],al ; TURN OFF MOTOR RUNNING BITS
1005 000006FD 2005[A6380100] and [MOTOR_STATUS], al
1006 ;and [ORG_MOTOR_STATUS], al
1007 00000703 B00C MOV AL,0CH ; bit 3 = enable IRQ & DMA,
1008 ; bit 2 = enable controller
1009 ; 1 = normal operation
1010 ; 0 = reset
1011 ; bit 0, 1 = drive select
1012 ; bit 4-7 = motor running bits
1013 00000705 66BAF203 MOV DX,03F2H ; FDC CTL PORT
1014 00000709 EE OUT DX,AL ; TURN OFF THE MOTOR
1015
1016 T6: ;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 E822EC0000 call clock ; Multi Tasking control procedure
1024
T7:

```

```

1025 ; 14/10/2015
1026 MOV AL,EOI ; GET END OF INTERRUPT MASK
1027 CLI ; DISABLE INTERRUPTS TILL STACK CLEARED
1028 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 cmp byte [u.t_lock], 0 ; T_LOCK
1035 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 cmp byte [u.r_lock], 0 ; R_LOCK, IRQ callback service lock !
1040 ja short timer_int_return ; Timer Lock : 'sysrele' is needed !
1041 ; 15/01/2017
1042 cmp byte [priority], 2
1043 jnb short T8 ; current process has a timer event (15/01/2017)
1044 ; 22/05/2016
1045 cmp byte [p_change], 0 ; in 'set_run_sequence', in 'rtc_p'
1046 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 xor eax, eax ; 26/12/2016
1053 mov [p_change], al ; 0
1054 ;mov byte [priority], 2 ; 15/01/2017 (there is a timer event)
1055 ;
1056 cmp byte [sysflg], 0FFh ; user or system space ?
1057 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 mov [u.quant], al ; 0
1064 ;
1065 timer_int_return: ; 23/05/2016 - jump from 'rtc_int' ('rtc_int_2')
1066 mov ecx, [cr3reg] ; previous value/content of cr3 register
1067 mov cr3, ecx ; restore cr3 register content
1068 ;
1069 popad ; edi, esi, ebp, temp (increment esp by 4), ebx, edx, ecx, eax
1070 ;
1071 pop gs
1072 pop fs
1073 pop es
1074 pop ds
1075 ;
1076 iretd ; return from interrupt
1077 ;
1078 rtc_int_3:
1079 inc byte [sysflg] ; now, we are in system space
1080 ;
1081 jmp sysrelease ; change running process immediatelly
1082 ;
1083 T8:
1084 ; 13/01/2017 (eax -> ebx)
1085 ; callback checking... (19/12/2016)
1086 xor ebx, ebx
1087 xchg ebx, [u.tcb] ; callback address (0 = normal return)
1088 or ebx, ebx
1089 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 inc byte [u.t_lock]
1098 mov cl, [sysflg]
1099 mov [u.t_mode], cl
1100 ;
1101 mov ebp, [tss.esp0] ; kernel stack address (for ring 0)
1102 sub ebp, 20 ; eip, cs, eflags, esp, ss
1103 mov [u.sp], ebp
1104 mov [u.usp], esp
1105 ;
1106 ;or word [ebp+8], 200h ; 22/01/2017, force enabling interrupts
1107 ;
1108 mov eax, [esp+28] ; pushed eax
1109 mov [u.r0], eax
1110 ;
1111 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 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 mov eax, [ebp+16] ; SS (UDATA)
1126 mov esi, esp

```

```

1127 000007BB 50      push    eax
1128 000007BC 50      push    eax
1129 000007BD 89E7     mov     edi, esp
1130 000007BF 893D[A84E0100]     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[A44E0100]     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[004F0100]     mov     ecx, [u.pgdir]
1141 000007E2 890D[A44F0100]     mov     [cr3reg], ecx
1142
1143      ; 13/01/2017 (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' system 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[A44E0100]     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[9E380100]  mov     ah, [RTC_2Hz] ; 2 Hz interrupt to 1 Hz function
1249 00000804 80F401      xor     ah, 1
1250 00000807 8825[9E380100]  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[A44F0100]  mov     [cr3reg], ecx ; save current cr3 register value/content
1274      ;
1275 00000829 803D[1C4F0100]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[20380100]  cmp     ecx, [k_page_dir]
1280 00000838 7409      je      short rtc_int_1
1281      ;
1282 0000083A 8B0D[20380100]  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 - dsctpm.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
1322      ; //////////////////////////////////
1323
1324      ; 28/08/2014
1325      irq0:
1326 00000865 6A00      push     dword 0
1327 00000867 EB48      jmp     short which_irq
1328      irq1:
1329 00000869 6A01      push     dword 1
1330 0000086B EB44      jmp     short which_irq

```

```
1331
1332 0000086D 6A02
1333 0000086F EB40
1334
1335
1336
1337 00000871 2EFF15[D6F50000]
1338 00000878 6A03
1339 0000087A EB35
1340
1341
1342
1343 0000087C 2EFF15[D2F50000]
1344 00000883 6A04
1345 00000885 EB2A
1346
1347 00000887 6A05
1348 00000889 EB26
1349
1350 0000088B 6A06
1351 0000088D EB22
1352
1353 0000088F 6A07
1354 00000891 EB1E
1355
1356 00000893 6A08
1357 00000895 EB1A
1358
1359 00000897 6A09
1360 00000899 EB16
1361
1362 0000089B 6A0A
1363 0000089D EB12
1364
1365 0000089F 6A0B
1366 000008A1 EB0E
1367
1368 000008A3 6A0C
1369 000008A5 EB0A
1370
1371 000008A7 6A0D
1372 000008A9 EB06
1373
1374 000008AB 6A0E
1375 000008AD EB02
1376
1377 000008AF 6A0F
1378
1379
1380
1381
1382
1383
1384
1385 000008B1 870424
1386 000008B4 53
1387 000008B5 56
1388 000008B6 57
1389 000008B7 1E
1390 000008B8 06
1391
1392 000008B9 88C3
1393
1394 000008BB B810000000
1395 000008C0 8ED8
1396 000008C2 8EC0
1397
1398 000008C4 FC
1399
1400 000008C5 8105[EC030100]A000-
1400 000008CD 0000
1401
1402 000008CF B417
1403
1404 000008D1 8B3D[EC030100]
1405 000008D7 B049
1406 000008D9 66AB
1407 000008DB B052
1408 000008DD 66AB
1409 000008DF B051
1410 000008E1 66AB
1411 000008E3 B020
1412 000008E5 66AB
1413 000008E7 88D8
1414 000008E9 3C0A
1415 000008EB 7208
1416 000008ED B031
1417 000008EF 66AB
1418 000008F1 88D8
1419 000008F3 2C0A
1420
1421 000008F5 0430
1422 000008F7 66AB
1423 000008F9 B020
1424 000008FB 66AB
1425 000008FD B021
1426 000008FF 66AB
1427 00000901 B020
1428 00000903 66AB
1429
1430 00000905 80FB07
1431 00000908 7604

irq2:
    push        dword 2
    jmp         short which_irq
irq3:
    ; 20/11/2015
    ; 24/10/2015
    call        dword [cs:com2_irq3]
    push        dword 3
    jmp         short which_irq
irq4:
    ; 20/11/2015
    ; 24/10/2015
    call        dword [cs:com1_irq4]
    push        dword 4
    jmp         short which_irq
irq5:
    push        dword 5
    jmp         short which_irq
irq6:
    push        dword 6
    jmp         short which_irq
irq7:
    push        dword 7
    jmp         short which_irq
irq8:
    push        dword 8
    jmp         short which_irq
irq9:
    push        dword 9
    jmp         short which_irq
irq10:
    push        dword 10
    jmp         short which_irq
irq11:
    push        dword 11
    jmp         short which_irq
irq12:
    push        dword 12
    jmp         short which_irq
irq13:
    push        dword 13
    jmp         short which_irq
irq14:
    push        dword 14
    jmp         short which_irq
irq15:
    push        dword 15
    ; jmp         short which_irq

    ; 22/01/2017
    ; 19/10/2015
    ; 29/08/2014
    ; 21/08/2014
which_irq:
    xchg        eax, [esp] ; 28/08/2014
    push        ebx
    push        esi
    push        edi
    push        ds
    push        es
    ;
    mov         bl, al
    ;
    mov         eax, KDATA
    mov         ds, ax
    mov         es, ax
    ; 19/10/2015
    cld
    ; 27/08/2014
    add         dword [scr_row], 0A0h

    ;
    mov         ah, 17h ; blue (1) background,
                        ; light gray (7) forecolor
    mov         edi, [scr_row]
    mov         al, 'I'
    stosw
    mov         al, 'R'
    stosw
    mov         al, 'Q'
    stosw
    mov         al, ' '
    stosw
    mov         al, bl
    cmp         al, 10
    jb          short ii1
    mov         al, 'l'
    stosw
    mov         al, bl
    sub         al, 10
ii1:
    add         al, '0'
    stosw
    mov         al, ' '
    stosw
    mov         al, '!'
    stosw
    mov         al, ' '
    stosw
    ; 23/02/2015
    cmp         bl, 7 ; check for IRQ 8 to IRQ 15
    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
1436 0000090E B020                      ii2:    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
1455 00000917 6A00                      exc0:    push     dword 0
1456 00000919 E990000000                jmp     cpu_except
1457
1458 0000091E 6A01                      exc1:    push     dword 1
1459 00000920 E989000000                jmp     cpu_except
1460
1461 00000925 6A02                      exc2:    push     dword 2
1462 00000927 E982000000                jmp     cpu_except
1463
1464 0000092C 6A03                      exc3:    push     dword 3
1465 0000092E EB7E                      jmp     cpu_except
1466
1467 00000930 6A04                      exc4:    push     dword 4
1468 00000932 EB7A                      jmp     cpu_except
1469
1470 00000934 6A05                      exc5:    push     dword 5
1471 00000936 EB76                      jmp     cpu_except
1472
1473 00000938 6A06                      exc6:    push     dword 6
1474 0000093A EB72                      jmp     cpu_except
1475
1476 0000093C 6A07                      exc7:    push     dword 7
1477 0000093E EB6E                      jmp     cpu_except
1478
1479                                     exc8:    ; [esp] = Error code
1480 00000940 6A08                      push     dword 8
1481 00000942 EB5C                      jmp     cpu_except_en
1482
1483 00000944 6A09                      exc9:    push     dword 9
1484 00000946 EB66                      jmp     cpu_except
1485
1486                                     exc10:   ; [esp] = Error code
1487 00000948 6A0A                      push     dword 10
1488 0000094A EB54                      jmp     cpu_except_en
1489
1490                                     exc11:   ; [esp] = Error code
1491 0000094C 6A0B                      push     dword 11
1492 0000094E EB50                      jmp     cpu_except_en
1493
1494                                     exc12:   ; [esp] = Error code
1495 00000950 6A0C                      push     dword 12
1496 00000952 EB4C                      jmp     cpu_except_en
1497
1498                                     exc13:   ; [esp] = Error code
1499 00000954 6A0D                      push     dword 13
1500 00000956 EB48                      jmp     cpu_except_en
1501
1502                                     exc14:   ; [esp] = Error code
1503 00000958 6A0E                      push     dword 14
1504 0000095A EB44                      jmp     short cpu_except_en
1505
1506 0000095C 6A0F                      exc15:   push     dword 15
1507 0000095E EB4E                      jmp     cpu_except
1508
1509 00000960 6A10                      exc16:   push     dword 16
1510 00000962 EB4A                      jmp     cpu_except
1511
1512                                     exc17:   ; [esp] = Error code
1513 00000964 6A11                      push     dword 17
1514 00000966 EB38                      jmp     short cpu_except_en
1515
1516 00000968 6A12                      exc18:   push     dword 18
1517 0000096A EB42                      jmp     short cpu_except
1518
1519 0000096C 6A13                      exc19:   push     dword 19
1520 0000096E EB3E                      jmp     short cpu_except
1521
1522 00000970 6A14                      exc20:   push     dword 20
1523 00000972 EB3A                      jmp     short cpu_except
1524
1525 00000974 6A15                      exc21:   push     dword 21
1526 00000976 EB36                      jmp     short cpu_except
1527
1528 00000978 6A16                      exc22:   push     dword 22
1529 0000097A EB32                      jmp     short cpu_except
1530
1531 0000097C 6A17                      exc23:   push     dword 23
1532 0000097E EB2E                      jmp     short cpu_except
1533
1534                                     exc24:

```

```

1534 00000980 6A18          push     dword 24
1535 00000982 EB2A          jmp      short cpu_except
1536
exc25:          push     dword 25
          jmp      short cpu_except
1537 00000984 6A19          push     dword 26
1538 00000986 EB26          jmp      short cpu_except
1539
exc26:          push     dword 27
          jmp      short cpu_except
1540 00000988 6A1A          push     dword 28
1541 0000098A EB22          jmp      short cpu_except
1542
exc27:          push     dword 29
          jmp      short cpu_except
1543 0000098C 6A1B          push     dword 30
1544 0000098E EB1E          jmp      short cpu_except
1545
exc28:          push     dword 31
          jmp      short cpu_except
1546 00000990 6A1C          push     dword 32
1547 00000992 EB1A          jmp      short cpu_except
1548
exc29:          push     dword 33
          jmp      short cpu_except
1549 00000994 6A1D          push     dword 34
1550 00000996 EB16          jmp      short cpu_except
1551
exc30:          push     dword 35
          jmp      short cpu_except_en
1552 00000998 6A1E          push     dword 36
1553 0000099A EB04          jmp      short cpu_except_en
1554
exc31:          push     dword 37
          jmp      short cpu_except
1555
          ; 19/10/2015
1558          ; 19/09/2015
1559          ; 01/09/2015
1560          ; 28/08/2015
1561          ; 28/08/2014
1562
cpu_except_en:
1564 000009A0 87442404      xchg     eax, [esp+4] ; Error code
1565 000009A4 36A3[C0500100] 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
          cld
1581 000009AE FC      xchg     eax, [esp]
1582 000009AF 870424      ; eax = Exception number
1583                                     ; [esp] = eax (before exception)
1584
          push     ebx
1585 000009B2 53      push     esi
1586 000009B3 56      push     edi
1587 000009B4 57      push     ds
1588 000009B5 1E      push     es
1589 000009B6 06      ; 28/08/2015
1590                                     mov     bx, KDATA
1591 000009B7 66BB1000   mov     ds, bx
1592 000009BB 8EDB      mov     es, bx
1593 000009BD 8EC3      mov     ebx, cr3
1594 000009BF 0F20DB   push    ebx ; (*) page directory
1595 000009C2 53      ; 19/10/2015
1596                                     cld
1597 000009C3 FC      ; 25/03/2015
1598                                     mov     ebx, [k_page_dir]
1599 000009C4 8B1D[20380100] mov     cr3, ebx
1600 000009CA 0F22DB   ; 28/08/2015
1601                                     cmp     eax, 0Eh ; 14, PAGE FAULT
1602 000009CD 83F80E      jne     short cpu_except_nfp
1603 000009D0 750F      call    page_fault_handler
1604 000009D2 E87A440000 and     eax, eax
1605 000009D7 21C0      jz      iiretp ; 01/09/2015
1606 000009D9 0F8401010000 mov     al, 0Eh ; 14
1607 000009DF B00E
1608
cpu_except_nfp:
1609          ; 23/08/2016
1610 000009E1 803D[225E0000]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[C4500100] 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 hlok
1630 00000A1A 0407      add     al, 'A'-' ':'
1631
hlok:
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
1637 00000A27 86E0
1638 00000A29 66053030
1639 00000A2D 66A3[44060100]
1640
1641
1642 00000A33 A1[C4500100]
1643 00000A38 51
1644 00000A39 52
1645 00000A3A 89E3
1646
1647 00000A3C B910000000
1648
1649
1650
1651
1652 00000A41 31D2
1653 00000A43 F7F1
1654 00000A45 6652
1655 00000A47 39C8
1656 00000A49 73F6
1657 00000A4B BF[4F060100]
1658
1659
1660 00000A50 89C2
1661
1662
1663 00000A52 8A82[1B330000]
1664 00000A58 AA
1665 00000A59 39E3
1666 00000A5B 7606
1667 00000A5D 6658
1668 00000A5F 88C2
1669 00000A61 EBEF
1670
1671 00000A63 B068
1672 00000A65 AA
1673 00000A66 B020
1674 00000A68 AA
1675 00000A69 30C0
1676 00000A6B AA
1677
1678 00000A6C 5A
1679 00000A6D 59
1680
1681 00000A6E B44F
1682
1683 00000A70 BE[34060100]
1684
1685
1686
1687 00000A75 8105[EC030100]A000-
1687 00000A7D 0000
1688 00000A7F 8B3D[EC030100]
1689
1690 00000A85 C605[A14E0100]00
1691 00000A8C FB
1692
1693 00000A8D E8EFFFBBBB
1694
1695 00000A92 B410
1696 00000A94 E87D010000
1697
1698 00000A99 B003
1699 00000A9B E8C50A0000
1700
1701 00000AA0 B801000000
1702 00000AA5 E9ECBA0000
1703
1704
1705
1706
1707
1708
1709
1710 00000AAA 50
1711 00000AAB 53
1712 00000AAC 56
1713 00000AAD 57
1714 00000AAE 1E
1715 00000AAF 06
1716
1717 00000AB0 66B81000
1718 00000AB4 8ED8
1719 00000AB6 8EC0
1720
1721 00000AB8 0F20D8
1722 00000ABB 50
1723
1724 00000ABC B467
1725
1726 00000ABE BE[FC040100]
1727
1728
1729 00000AC3 8105[EC030100]A000-
1729 00000ACB 0000
1730 00000ACD 8B3D[EC030100]
1731
1732 00000AD3 E8A9FBFFFF
1733
1734
1735 00000AD8 B020

h2ok:
    xchg  ah, al
    add   ax, '00'
    mov   [excnstr], ax
    ;
    ; 29/08/2014
    mov   eax, [FaultOffset]
    push  ecx
    push  edx
    mov   ebx, esp
    ; 28/08/2015
    mov   ecx, 16          ; divisor value to convert binary number
                          ; to hexadecimal string
    ;mov   ecx, 10          ; divisor to convert
                          ; binary number to decimal string

b2d1:
    xor   edx, edx
    div   ecx
    push  dx
    cmp   eax, ecx
    jnb   short b2d1
    mov   edi, EIPstr ; EIP value
                          ; points to instruction which faults
    ; 28/08/2015
    mov   edx, eax

b2d2:
    ;add   al, '0'
    mov   al, [edx+hexchrs]
    stosb          ; write hexadecimal digit to its place
    cmp   ebx, esp
    jna   short b2d3
    pop   ax
    mov   dl, al
    jmp   short b2d2

b2d3:
    mov   al, 'h' ; 28/08/2015
    stosb
    mov   al, 20h          ; space
    stosb
    xor   al, al          ; to do it an ASCII string
    stosb
    ;
    pop   edx
    pop   ecx
    ;
    mov   ah, 4Fh          ; red (4) background,
                          ; white (F) forecolor
    mov   esi, exc_msg ; message offset
    ;
    ; 20/01/2017 (!cpu exception!)
    ;
    add   dword [scr_row], 0A0h

    mov   edi, [scr_row]
    ;
    mov   byte [sysflg], 0 ; system mode
    sti
    ;
    call  printk
    ;
    mov   ah, 10h
    call  int16h ; getc
    ;
    mov   al, 3
    call  _set_mode
    ;
    mov   eax, 1
    jmp   sysexit ; terminate process !!!

    ; 22/01/2017
    ; 18/04/2016
    ; 28/08/2015
    ; 23/02/2015
    ; 20/08/2014

ignore_int:
    push  eax
    push  ebx ; 23/02/2015
    push  esi
    push  edi
    push  ds
    push  es
    ; 18/04/2016
    mov   ax, KDATA
    mov   ds, ax
    mov   es, ax
    ; 28/08/2015
    mov   eax, cr3
    push  eax ; (*) page directory
    ;
    mov   ah, 67h          ; brown (6) background,
                          ; light gray (7) forecolor
    mov   esi, int_msg ; message offset

piemsg:
    ; 27/08/2014
    add   dword [scr_row], 0A0h

    mov   edi, [scr_row]
    ;
    call  printk
    ;
    ; 23/02/2015
    mov   al, 20h          ; END OF INTERRUPT COMMAND TO

```

```

1736 00000ADA E6A0          out    0A0h, al ; the 2nd 8259
1737                        ; 22/08/2014
1738 00000ADC B020          mov     al, 20h ; END OF INTERRUPT COMMAND TO 8259
1739 00000ADE E620          out     20h, al      ; 8259 PORT
1740
1741                        iiretp:
1742                        ; 22/01/2017
1743                        ; 01/09/2015
1744                        ; 28/08/2015
1744 00000AE0 58             pop     eax ; (*) page directory
1745 00000AE1 0F22D8         mov     cr3, eax
1746
1747                        iiret:
1747                        pop     es
1748                        pop     ds
1749                        pop     edi
1750                        pop     esi
1751                        pop     ebx ; 29/08/2014
1752                        pop     eax
1753                        iretd
1754
1755                        ; 23/05/2016
1756                        ; 22/08/2014
1757                        ; IBM PC/AT BIOS source code ----- 10/06/85 (bios.asm)
1758                        ; (INT 1Ah)
1759                        ; Linux (v0.12) source code (main.c) by Linus Torvalds (1991)
1760
1760                        time_of_day:
1761                        call    UPD_IPR          ; WAIT TILL UPDATE NOT IN PROGRESS
1762                        jc      short time_of_day_retn ; 23/05/2016
1763                        mov     al, CMOS_SECONDS
1764                        call    CMOS_READ
1765                        mov     [time_seconds], al
1766                        mov     al, CMOS_MINUTES
1767                        call    CMOS_READ
1768                        mov     [time_minutes], al
1769                        mov     al, CMOS_HOURS
1770                        call    CMOS_READ
1771                        mov     [time_hours], al
1772                        mov     al, CMOS_DAY_WEEK
1773                        call    CMOS_READ
1774                        mov     [date_wday], al
1775                        mov     al, CMOS_DAY_MONTH
1776                        call    CMOS_READ
1777                        mov     [date_day], al
1778                        mov     al, CMOS_MONTH
1779                        call    CMOS_READ
1780                        mov     [date_month], al
1781                        mov     al, CMOS_YEAR
1782                        call    CMOS_READ
1783                        mov     [date_year], al
1784                        mov     al, CMOS_CENTURY
1785                        call    CMOS_READ
1786                        mov     [date_century], al
1787                        ;
1788                        mov     al, CMOS_SECONDS
1789                        call    CMOS_READ
1790                        cmp     al, [time_seconds]
1791                        jne     short time_of_day
1792
1793                        time_of_day_retn:
1794                        retn
1795
1796                        ; 15/01/2017
1797                        ; 10/06/2016
1798                        ; 07/06/2016
1799                        ; 06/06/2016
1800                        ; 23/05/2016
1801
1801                        rtc_p:
1802                        mov     cl, 1 ; 15/01/2017
1803                        jmp     short rtc_p0
1804
1804                        u_timer:
1805                        ; Timer Events with 18.2 Hz Timer Ticks
1806                        ; (and also timer events with RTC seconds)
1807                        sub     cl, cl ; mov cl, 0 ; 15/01/2017
1808
1808                        rtc_p0:
1809                        ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
1810                        ; Major Modification:
1811                        ; Check and Perform Timer Events (for RTC)
1812                        ; 25/08/2014 - 07/09/2014
1813                        ; Retro UNIX 386 v1:
1814                        ; Print Real Time Clock content
1815
1816                        ; 15/01/2017
1817                        mov     byte [priority], cl ; 0 or 1 (not 2)
1818                        mov     ch, [timer_events]
1819                        and     ch, ch
1820                        jz      short rtc_p3
1821
1822                        mov     esi, timer_set ; beginning address of
1823                        ; timer events space
1824
1824                        rtc_p1:
1825                        mov     eax, [esi]
1826                        and     al, al ; 0 = free, >0 = process no.
1827                        jz      short rtc_p4
1828                        ;
1829                        ror     eax, 16
1830                        ; ah = response value, al = interrupt type
1831                        ; 15/01/2017
1832                        ; cl = interrupt source
1833                        ; 1 = RTC, 0 = PIT
1834                        cmp     al, cl
1835                        jne     short rtc_p2 ; not as requested or undefined !
1836                        cmp     al, 1 ; 1 ; RTC interrupt ?
1837                        je      short rtc_p5 ; yes, check for response

```

```

1838 ; 06/06/2016 - 18.2 Hz Timer Ticks
1839 sub dword [esi+8], 10 ; 1 tick = 10
1840 jna short rtc_p6 ; continue for responding
1841 rtc_p2:
1842 ; 15/01/2017 (cl -> ch)
1843 ; 07/06/2016
1844 dec ch ; remain count of timer events
1845 jnz short rtc_p4
1846 rtc_p3:
1847 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 add esi, 16 ; next timer event
1852 jmp short rtc_p1
1853 rtc_p5:
1854 ; current timer count ; 06/06/2016 (182)
1855 sub dword [esi+8], 182 ; 1 second (10*18.2)
1856 ja short rtc_p2 ; check for the next
1857 rtc_p6:
1858 ; it is the time of response!
1859 mov ebx, [esi+4] ; set (count limit) value
1860 mov [esi+8], ebx ; reset count down value
1861 ; to count limit
1862 ; 19/12/2016
1863 ; 10/12/2016 - timer callback modification
1864 mov edi, [esi+12] ; response (or callback) address
1865 cmp byte [esi+1], 0 ; >0 = callback
1866 jna short rtc_p8
1867
1868 ; timer callback !
1869 movzx ebx, byte [esi] ; process number (>0)
1870 mov eax, ebx
1871 shl bl, 2 ; *4
1872 mov [ebx+p.tcb-4], edi ; user's callback service addr
1873 cmp al, [u.uno]
1874 jne short rtc_p9
1875 mov [u.tcb], edi
1876 rtc_p7:
1877 ; 15/01/2017
1878 mov al, 2
1879 mov [priority], al ; 2
1880 ; 10/01/2017
1881 ;mov byte [u.pri], 2
1882 mov [u.pri], al ; 2
1883 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 mov [edi], ah ; response value
1890 ;
1891 rol eax, 16
1892 ; 15/01/2017
1893 cmp al, [u.uno] ; running process ?
1894 je short rtc_p7
1895 rtc_p9:
1896 ; al = process number ; 10/06/2016
1897 mov dl, 2 ; priority, 2 = event (high)
1898 call set_run_sequence ; 19/05/2016
1899 jmp short rtc_p2 ; 10/06/2016
1900
1901 ; Default IRQ 7 handler against spurious IRQs (from master PIC)
1902 ; 25/02/2015 (source: http://wiki.osdev.org/8259\_PIC)
1903 default_irq7:
1904 push ax
1905 mov al, 0Bh ; In-Service register
1906 out 20h, al
1907 jmp short $+2
1908 jmp short $+2
1909 in al, 20h
1910 and al, 80h ; bit 7 (is it real IRQ 7 or fake?)
1911 jz short irq7_iret ; Fake (spurious) IRQ, do not send EOI
1912 mov al, 20h ; EOI
1913 out 20h, al
1914 irq7_iret:
1915 pop ax
1916 iretd
1917
1918 bcd_to_ascii:
1919 ; 25/08/2014
1920 ; INPUT ->
1921 ; al = Packed BCD number
1922 ; OUTPUT ->
1923 ; ax = ASCII word/number
1924 ;
1925 ;
1926 ; Erdogan Tan - 1998 (proc_hex) - TRDOS.ASM (2004-2011)
1927 ;
1928 db 0D4h, 10h ; Undocumented inst. AAM
1929 ; AH = AL / 10h
1930 ; AL = AL MOD 10h
1931 or ax, '00' ; Make it ASCII based
1932
1933 xchg ah, al
1934
1935 retn
1936
1937 %include 'keyboard.s' ; 07/03/2015
1938 <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     <1> ; pushfd ; 28/08/2014
40     <1> ; push cs
41     <1> ; call KEYBOARD_IO_1 ; getc_int
42     <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 (_kld)
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 :
95     <1> ; 09H 13.3 19H 3.3 :
96     <1> ; 0AH 12.0 1AH 3.0 :
97     <1> ; 0BH 10.9 1BH 2.7 :
98     <1> ; 0CH 10.0 1CH 2.5 :
99     <1> ; 0DH 9.2 1DH 2.3 :
100    <1> ; 0EH 8.6 1EH 2.1 :
101    <1> ; 0FH 8.0 1FH 2.0 :
102    <1> ; :
103    <1> ; (BH) = TYPAMATIC DELAY (BITS 2 - 7 MUST BE RESET TO 0) :

```

```
104 <1> ; :
105 <1> ; REGISTER DELAY :
106 <1> ; VALUE VALUE :
107 <1> ; ----- :
108 <1> ; 00H 250 ms :
109 <1> ; 01H 500 ms :
110 <1> ; 02H 750 ms :
111 <1> ; 03H 1000 ms :
112 <1> ;-----:
113 <1> ; (AH)= 05H PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD :
114 <1> ; BUFFER AS IF STRUCK FROM KEYBOARD :
115 <1> ; ENTRY: (CL) = ASCII CHARACTER :
116 <1> ; (CH) = SCAN CODE :
117 <1> ; EXIT: (AH) = 00H = SUCCESSFUL OPERATION :
118 <1> ; (AL) = 01H = UNSUCCESSFUL - BUFFER FULL :
119 <1> ; FLAGS: CARRY IF ERROR :
120 <1> ;-----:

121 <1> ; (AH)= 10H EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD, :
122 <1> ; OTHERWISE SAME AS FUNCTION AH=0 :
123 <1> ;-----:
124 <1> ; (AH)= 11H EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD, :
125 <1> ; OTHERWISE SAME AS FUNCTION AH=1 :
126 <1> ;-----:
127 <1> ; (AH)= 12H RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER :
128 <1> ; AL = BITS FROM KB_FLAG, AH = BITS FOR LEFT AND RIGHT :
129 <1> ; CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3 :
130 <1> ; OUTPUT :
131 <1> ; AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED :
132 <1> ; ALL REGISTERS RETAINED :
133 <1> ;-----:
134 <1>
135 <1> ; 15/01/2017
136 <1> ; 14/01/2017
137 <1> ; 02/01/2017
138 <1> ; 29/05/2016
139 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
140 <1> int32h: ; Keyboard BIOS
141 <1>
142 <1> KEYBOARD_IO_1:
143 <1> ;sti ; INTERRUPTS BACK ON
144 <1> ; 29/05/2016
145 00000C1E 80642408BE <1> and byte [esp+8], 10111110b ; clear zero flag and cary flag
146 <1> ;
147 00000C23 1E <1> push ds ; SAVE CURRENT DS
148 00000C24 53 <1> push ebx ; SAVE BX TEMPORARILY
149 <1> ;push ecx ; SAVE CX TEMPORARILY
150 00000C25 66BB1000 <1> mov bx, KDATA
151 00000C29 8EDB <1> mov ds, bx ; PUT SEGMENT VALUE OF DATA AREA INTO DS
152 <1>
153 <1> ; 14/01/2017
154 00000C2B 8B1C24 <1> mov ebx, [esp]
155 <1> ;; 15/01/2017
156 <1> ; 02/01/2017
157 <1> ;mov byte [intflg], 32h ; keyboard interrupt
158 00000C2E FB <1> sti
159 <1> ;
160 <1>
161 00000C2F 08E4 <1> or ah, ah ; CHECK FOR (AH)= 00H
162 00000C31 743A <1> jz short _K1 ; ASCII_READ
163 00000C33 FECC <1> dec ah ; CHECK FOR (AH)= 01H
164 00000C35 7453 <1> jz short _K2 ; ASCII_STATUS
165 00000C37 FECC <1> dec ah ; CHECK FOR (AH)= 02H
166 00000C39 0F8494000000 <1> jz _K3 ; SHIFT STATUS
167 00000C3F FECC <1> dec ah ; CHECK FOR (AH)= 03H
168 00000C41 0F8493000000 <1> jz _K300 ; SET TYPAMATIC RATE/DELAY
169 00000C47 80EC02 <1> sub ah, 2 ; CHECK FOR (AH)= 05H
170 00000C4A 0F84BC000000 <1> jz _K500 ; KEYBOARD WRITE
171 <1> _KIO1:
172 00000C50 80EC0B <1> sub ah, 11 ; AH = 10H
173 00000C53 740C <1> jz short _K1E ; EXTENDED ASCII READ
174 00000C55 FECC <1> dec ah ; CHECK FOR (AH)= 11H
175 00000C57 7422 <1> jz short _K2E ; EXTENDED_ASCII_STATUS
176 00000C59 FECC <1> dec ah ; CHECK FOR (AH)= 12H
177 00000C5B 7458 <1> jz short _K3E ; EXTENDED_SHIFT_STATUS
178 <1> _KIO_EXIT:
179 <1> ; 02/01/2017
180 00000C5D FA <1> cli
181 <1> ;mov byte [intflg], 0 ;; 15/01/2017
182 <1> ;
183 <1> ;pop ecx ; RECOVER REGISTER
184 00000C5E 5B <1> pop ebx ; RECOVER REGISTER
185 00000C5F 1F <1> pop ds ; RECOVER SEGMENT
186 00000C60 CF <1> iretd ; INVALID COMMAND, EXIT
187 <1>
188 <1> ;----- ASCII CHARACTER
189 <1> _K1E:
190 00000C61 E8D3000000 <1> call _K1S ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
191 00000C66 E848010000 <1> call _KIO_E_XLAT ; ROUTINE TO XLATE FOR EXTENDED CALLS
192 00000C6B EBF0 <1> jmp short _KIO_EXIT ; GIVE IT TO THE CALLER
193 <1> _K1:
194 00000C6D E8C7000000 <1> call _K1S ; GET A CHARACTER FROM THE BUFFER
195 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' instruction 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[EE5D0000] <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[EE5D0000] <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[F05D0000] <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[ED5D0000] <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[FE5D0000] <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[FA5D0000] <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[FE5D0000] <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[FA5D0000] <1>      mov     ebx, [BUFFER_HEAD] ; GET POINTER TO HEAD OF BUFFER
324 00000D40 3B1D[FE5D0000] <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:
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[FA5D0000] <1>      mov     ebx, [BUFFER_HEAD] ; GET POINTER TO HEAD OF BUFFER
339 00000D51 3B1D[FE5D0000] <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[EF5D0000] <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[FA5D0000] <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[FA5D0000] <1>      mov     ebx, [BUFFER_HEAD] ; GET HEAD POINTER
364 00000D8B 3B1D[FE5D0000] <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[EF5D0000] <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, MESSAGE A BIT
396 00000DC7 3C0A <1>      cmp     al, 0Ah          ; CTRL KEYPAD ENTER CODE?
397 00000DC9 7404 <1>      je      short _KIO_S1        ; YES, MESSAGE THE SAME
398 00000DCB B435 <1>      mov     ah, 35h          ; NO, MUST BE KEYPAD /
399 <1>      _KIO_RET: ; 03/12/2014
400 00000DCD F8 <1>      clc     ;
401 00000DCE C3 <1>      retn     ;
402 <1>      ;jmp     short _KIO_USE        ; GIVE TO CALLER
403 <1>      _KIO_S1:
404 00000DCF B41C <1>      mov     ah, 1Ch          ; CONVERT TO COMPATIBLE OUTPUT
405 <1>      ;jmp     short _KIO_USE        ; GIVE TO CALLER
406 00000DD1 C3 <1>      retn     ;
407 <1>      _KIO_S2:
408 00000DD2 80FC84 <1>      cmp     ah, 84h          ; IS IT ONE OF EXTENDED ONES?

```



```

409 00000DD5 7715      <1>      ja      short _KIO_DIS          ; YES, THROW AWAY AND GET ANOTHER CHAR
410 00000DD7 3CF0      <1>      cmp      al, 0F0h          ; IS IT ONE OF THE FILL-INS?
411 00000DD9 7506      <1>      jne      short _KIO_S3          ; NO, TRY LAST TEST
412 00000DDB 08E4      <1>      or       ah, ah          ; AH = 0 IS SPECIAL CASE
413 00000DDD 740C      <1>      jz       short _KIO_USE          ; PASS THIS ON UNCHANGED
414 00000DDF EB0B      <1>      jmp      short _KIO_DIS          ; THROW AWAY THE REST
415                      <1>      _KIO_S3:
416 00000DE1 3CE0      <1>      cmp      al, 0E0h          ; IS IT AN EXTENSION OF A PREVIOUS ONE?
417                      <1>      ;jne      short _KIO_USE          ; NO, MUST BE A STANDARD CODE
418 00000DE3 75E8      <1>      jne      short _kio_ret
419 00000DE5 08E4      <1>      or       ah, ah          ; AH = 0 IS SPECIAL CASE
420 00000DE7 7402      <1>      jz       short _KIO_USE          ; JUMP IF AH = 0
421 00000DE9 30C0      <1>      xor      al, al          ; CONVERT TO COMPATIBLE OUTPUT
422                      <1>      ;jmp      short _KIO_USE          ; PASS IT ON TO CALLER
423                      <1>      _KIO_USE:
424                      <1>      ;clc          ; CLEAR CARRY TO INDICATE GOOD CODE
425 00000DEB C3          <1>      retn          ; RETURN
426                      <1>      _KIO_DIS:
427 00000DEC F9          <1>      stc          ; SET CARRY TO INDICATE DISCARD CODE
428 00000DED C3          <1>      retn          ; RETURN
429                      <1>
430                      <1>      ;----- INCREMENT BUFFER POINTER ROUTINE -----
431                      <1>      _K4:
432 00000DEE 43          <1>      inc      ebx
433 00000DEF 43          <1>      inc      ebx          ; MOVE TO NEXT WORD IN LIST
434 00000DF0 3B1D[F65D0000] <1>      cmp      ebx, [BUFFER_END]      ; AT END OF BUFFER?
435                      <1>      ;jne      short _K5          ; NO, CONTINUE
436 00000DF6 7206      <1>      jb       short _K5
437 00000DF8 8B1D[F25D0000] <1>      mov      ebx, [BUFFER_START]      ; YES, RESET TO BUFFER BEGINNING
438                      <1>      _K5:
439 00000DFE C3          <1>      retn
440                      <1>
441                      <1>      ; 20/02/2015
442                      <1>      ; 05/12/2014
443                      <1>      ; 26/08/2014
444                      <1>      ; KEYBOARD (HARDWARE) INTERRUPT - IRQ LEVEL 1
445                      <1>      ; (INT_09h - Retro UNIX 8086 v1 - U9.ASM, 07/03/2014)
446                      <1>      ;
447                      <1>      ; Derived from "KB_INT_1" procedure of IBM "pc-at"
448                      <1>      ; rombios source code (06/10/1985)
449                      <1>      ; 'keybd.asm', HARDWARE INT 09h - (IRQ Level 1)
450                      <1>
451                      <1>      ; EQUATES (IBM PC-XT-286 BIOS, 1986, 'POSQEQU.INC')
452                      <1>
453                      <1>      ;----- 8042 COMMANDS -----
454                      <1>      ENA_KBD      equ      0AEh          ; ENABLE KEYBOARD COMMAND
455                      <1>      DIS_KBD      equ      0ADh          ; DISABLE KEYBOARD COMMAND
456                      <1>      SHUT_CMD      equ      0FEh          ; CAUSE A SHUTDOWN COMMAND
457                      <1>      ;----- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----
458                      <1>      STATUS_PORT equ      064h          ; 8042 STATUS PORT
459                      <1>      INPT_BUF_FULL equ      00000010b      ; 1 = +INPUT BUFFER FULL
460                      <1>      PORT_A      equ      060h          ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
461                      <1>      ;----- 8042 KEYBOARD RESPONSE -----
462                      <1>      KB_ACK      equ      0FAh          ; ACKNOWLEDGE PROM TRANSMISSION
463                      <1>      KB_RESEND equ      0FEh          ; RESEND REQUEST
464                      <1>      KB_OVER_RUN equ      0FFh          ; OVER RUN SCAN CODE
465                      <1>      ;----- KEYBOARD/LED COMMANDS -----
466                      <1>      KB_ENABLE equ      0F4h          ; KEYBOARD ENABLE
467                      <1>      LED_CMD      equ      0EDh          ; LED WRITE COMMAND
468                      <1>      KB_TYPA_RD equ      0F3h          ; TYPAMATIC RATE/DELAY COMMAND
469                      <1>      ;----- KEYBOARD SCAN CODES -----
470                      <1>      NUM_KEY      equ      69          ; SCAN CODE FOR NUMBER LOCK KEY
471                      <1>      SCROLL_KEY equ      70          ; SCAN CODE FOR SCROLL LOCK KEY
472                      <1>      ALT_KEY      equ      56          ; SCAN CODE FOR ALTERNATE SHIFT KEY
473                      <1>      CTL_KEY      equ      29          ; SCAN CODE FOR CONTROL KEY
474                      <1>      CAPS_KEY      equ      58          ; SCAN CODE FOR SHIFT LOCK KEY
475                      <1>      DEL_KEY      equ      83          ; SCAN CODE FOR DELETE KEY
476                      <1>      INS_KEY      equ      82          ; SCAN CODE FOR INSERT KEY
477                      <1>      LEFT_KEY      equ      42          ; SCAN CODE FOR LEFT SHIFT
478                      <1>      RIGHT_KEY equ      54          ; SCAN CODE FOR RIGHT SHIFT
479                      <1>      SYS_KEY      equ      84          ; SCAN CODE FOR SYSTEM KEY
480                      <1>      ;----- ENHANCED KEYBOARD SCAN CODES -----
481                      <1>      ID_1      equ      0ABh          ; 1ST ID CHARACTER FOR KBX
482                      <1>      ID_2      equ      041h          ; 2ND ID CHARACTER FOR KBX
483                      <1>      ID_2A      equ      054h          ; ALTERNATE 2ND ID CHARACTER FOR KBX
484                      <1>      F11_M      equ      87          ; F11 KEY MAKE
485                      <1>      F12_M      equ      88          ; F12 KEY MAKE
486                      <1>      MC_E0      equ      224          ; GENERAL MARKER CODE
487                      <1>      MC_E1      equ      225          ; PAUSE KEY MARKER CODE
488                      <1>      ;----- FLAG EQUATES WITHIN @KB_FLAG-----
489                      <1>      RIGHT_SHIFT equ      00000001b      ; RIGHT SHIFT KEY DEPRESSED
490                      <1>      LEFT_SHIFT equ      00000010b      ; LEFT SHIFT KEY DEPRESSED
491                      <1>      CTL_SHIFT equ      00000100b      ; CONTROL SHIFT KEY DEPRESSED
492                      <1>      ALT_SHIFT equ      00001000b      ; ALTERNATE SHIFT KEY DEPRESSED
493                      <1>      SCROLL_STATE equ      00010000b      ; SCROLL LOCK STATE IS ACTIVE
494                      <1>      NUM_STATE equ      00100000b      ; NUM LOCK STATE IS ACTIVE
495                      <1>      CAPS_STATE equ      01000000b      ; CAPS LOCK STATE IS ACTIVE
496                      <1>      INS_STATE equ      10000000b      ; INSERT STATE IS ACTIVE
497                      <1>      ;----- FLAG EQUATES WITHIN @KB_FLAG_1 -----
498                      <1>      L_CTL_SHIFT equ      00000001b      ; LEFT CTL KEY DOWN
499                      <1>      L_ALT_SHIFT equ      00000010b      ; LEFT ALT KEY DOWN
500                      <1>      SYS_SHIFT equ      00000100b      ; SYSTEM KEY DEPRESSED AND HELD
501                      <1>      HOLD_STATE equ      00001000b      ; SUSPEND KEY HAS BEEN TOGGLED
502                      <1>      SCROLL_SHIFT equ      00010000b      ; SCROLL LOCK KEY IS DEPRESSED
503                      <1>      NUM_SHIFT equ      00100000b      ; NUM LOCK KEY IS DEPRESSED
504                      <1>      CAPS_SHIFT equ      01000000b      ; CAPS LOCK KEY IS DEPRESSED
505                      <1>      INS_SHIFT equ      10000000b      ; INSERT KEY IS DEPRESSED
506                      <1>      ;----- FLAGS EQUATES WITHIN @KB_FLAG_2 -----
507                      <1>      KB_LEDS      equ      00000111b      ; KEYBOARD LED STATE BITS
508                      <1>      ;          equ      00000001b      ; SCROLL LOCK INDICATOR
509                      <1>      ;          equ      00000010b      ; NUM LOCK INDICATOR
510                      <1>      ;          equ      00000100b      ; CAPS LOCK INDICATOR

```



```

511 <1> ; equ 00001000b ; RESERVED (MUST BE ZERO)
512 <1> KB_FA equ 00010000b ; ACKNOWLEDGMENT RECEIVED
513 <1> KB_FE equ 00100000b ; RESEND RECEIVED FLAG
514 <1> KB_PR_LED equ 01000000b ; MODE INDICATOR UPDATE
515 <1> KB_ERR equ 10000000b ; KEYBOARD TRANSMIT ERROR FLAG
516 <1> ;----- FLAGS EQUATES WITHIN @KB_FLAG_3 -----
517 <1> LC_E1 equ 00000001b ; LAST CODE WAS THE E1 HIDDEN CODE
518 <1> LC_E0 equ 00000010b ; LAST CODE WAS THE E0 HIDDEN CODE
519 <1> R_CTL_SHIFT equ 00000100b ; RIGHT CTL KEY DOWN
520 <1> R_ALT_SHIFT equ 00001000b ; RIGHT ALT KEY DOWN
521 <1> GRAPH_ON equ 00001000b ; ALT GRAPHICS KEY DOWN (WT ONLY)
522 <1> KBX equ 00010000b ; ENHANCED KEYBOARD INSTALLED
523 <1> SET_NUM_LK equ 00100000b ; FORCE NUM LOCK IF READ ID AND KBX
524 <1> LC_AB equ 01000000b ; LAST CHARACTER WAS FIRST ID CHARACTER
525 <1> RD_ID equ 10000000b ; DOING A READ ID (MUST BE BIT0)
526 <1> ;
527 <1> ;----- INTERRUPT EQUATES -----
528 <1> EOI equ 020h ; END OF INTERRUPT COMMAND TO 8259
529 <1> INTA00 equ 020h ; 8259 PORT
530 <1>
531 <1>
532 <1> kb_int:
533 <1>
534 <1> ; 17/10/2015 ('ctrlbrk')
535 <1> ; 05/12/2014
536 <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)
537 <1> ; 26/08/2014
538 <1> ;
539 <1> ; 03/06/86 KEYBOARD BIOS
540 <1> ;
541 <1> ;--- HARDWARE INT 09H -- (IRQ LEVEL 1) -----
542 <1> ;
543 <1> ; KEYBOARD INTERRUPT ROUTINE
544 <1> ;
545 <1> ;-----
546 <1>
547 <1> KB_INT_1:
548 00000DFF FB <1> sti ; ENABLE INTERRUPTS
549 <1> ;push ebp
550 00000E00 50 <1> push eax
551 00000E01 53 <1> push ebx
552 00000E02 51 <1> push ecx
553 00000E03 52 <1> push edx
554 00000E04 56 <1> push esi
555 00000E05 57 <1> push edi
556 00000E06 1E <1> push ds
557 00000E07 06 <1> push es
558 00000E08 FC <1> cld ; FORWARD DIRECTION
559 00000E09 66B81000 <1> mov ax, KDATA
560 00000E0D 8ED8 <1> mov ds, ax
561 00000E0F 8EC0 <1> mov es, ax
562 <1> ;
563 <1> ;---- WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
564 00000E11 B0AD <1> mov al, DIS_KBD ; DISABLE THE KEYBOARD COMMAND
565 00000E13 E8A9050000 <1> call SHIP_IT ; EXECUTE DISABLE
566 00000E18 FA <1> cli ; DISABLE INTERRUPTS
567 00000E19 B900000100 <1> mov ecx, 10000h ; SET MAXIMUM TIMEOUT
568 <1> KB_INT_01:
569 00000E1E E464 <1> in al, STATUS_PORT ; READ ADAPTER STATUS
570 00000E20 A802 <1> test al, INPT_BUF_FULL ; CHECK INPUT BUFFER FULL STATUS BIT
571 00000E22 E0FA <1> loopnz KB_INT_01 ; WAIT FOR COMMAND TO BE ACCEPTED
572 <1> ;
573 <1> ;---- READ CHARACTER FROM KEYBOARD INTERFACE
574 00000E24 E460 <1> in al, PORT_A ; READ IN THE CHARACTER
575 <1> ;
576 <1> ;---- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INT LEVEL 9H)
577 <1> ;MOV AH, 04FH ; SYSTEM INTERCEPT - KEY CODE FUNCTION
578 <1> ;STC ; SET CY=1 (IN CASE OF IRET)
579 <1> ;INT 15H ; CASSETTE CALL (AL)=KEY SCAN CODE
580 <1> ; ; RETURNS CY=1 FOR INVALID FUNCTION
581 <1> ;JC KB_INT_02 ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
582 <1> ;JMP K26 ; EXIT IF SYSTEM HANDLES SCAN CODE
583 <1> ; ; EXIT HANDLES HARDWARE EOI AND ENABLE
584 <1> ;
585 <1> ;---- CHECK FOR A RESEND COMMAND TO KEYBOARD
586 <1> KB_INT_02: ; (AL)= SCAN CODE
587 00000E26 FB <1> sti ; ENABLE INTERRUPTS AGAIN
588 00000E27 3CFE <1> cmp al, KB_RESEND ; IS THE INPUT A RESEND
589 00000E29 7411 <1> je short KB_INT_4 ; GO IF RESEND
590 <1> ;
591 <1> ;---- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD
592 00000E2B 3CFA <1> cmp al, KB_ACK ; IS THE INPUT AN ACKNOWLEDGE
593 00000E2D 751A <1> jne short KB_INT_2 ; GO IF NOT
594 <1> ;
595 <1> ;---- A COMMAND TO THE KEYBOARD WAS ISSUED
596 00000E2F FA <1> cli ; DISABLE INTERRUPTS
597 00000E30 800D[EF5D0000]10 <1> or byte [KB_FLAG_2], KB_FA ; INDICATE ACK RECEIVED
598 00000E37 E97A020000 <1> jmp K26 ; RETURN IF NOT (ACK RETURNED FOR DATA)
599 <1> ;
600 <1> ;---- RESEND THE LAST BYTE
601 <1> KB_INT_4:
602 00000E3C FA <1> cli ; DISABLE INTERRUPTS
603 00000E3D 800D[EF5D0000]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[EF5D0000] <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[F05D0000] <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[F05D0000]40 <1>      or      byte [KB_FLAG_3], LC_AB ; INDICATE 1ST ID WAS OK
636                                <1> RST_RD_ID:
637 00000E86 8025[F05D0000]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[F05D0000]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[ED5D0000]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[F05D0000]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[F05D0000]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[F05D0000]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[DA5C0000] <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[D85C0000] <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[EE5D0000]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[EE5D0000]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[EE5D0000]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[EE5D0000]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                ; INCONRE 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[ED5D0000]           <1>      mov    bl, [KB_FLAG]          ; PUT STATE FLAGS IN BL
748 00000F7A BF[D45C0000]             <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[D55C0000]           <1>      sub     edi, _K6+1          ; ADJUST PTR TO SCAN CODE MATCH
757 00000F94 8AA7[DC5C0000]           <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[ED5D0000]           <1>      or      [KB_FLAG], ah      ; TURN ON SHIFT BIT
769 00000FAF A80C                     <1>      test    al, 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[F05D0000]           <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[EE5D0000]           <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:
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:
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:
805 00001000 F6C303                   <1>      test    bl, LEFT_SHIFT+RIGHT_SHIFT
806 00001003 74F4                     <1>      jz      short K20          ; IS NUMERIC, STD. PROC.
807                                     <1>      ;
808                                     <1>      K22:
809 00001005 8425[EE5D0000]           <1>      test    ah, [KB_FLAG_1]    ; IS KEY ALREADY DEPRESSED
810 0000100B 0F85A5000000             <1>      jnz     K26                ; JUMP IF KEY ALREADY DEPRESSED
811                                     <1>      K22A:
812 00001011 0825[EE5D0000]           <1>      or      [KB_FLAG_1], ah    ; INDICATE THAT THE KEY IS DEPRESSED
813 00001017 3025[ED5D0000]           <1>      xor     [KB_FLAG], ah      ; TOGGLE THE SHIFT STATE
814                                     <1>      ;
815                                     <1>      ;----- TOGGLE LED IF CAPS, NUM OR SCROLL KEY DEPRESSED
816 0000101D F6C470                   <1>      test    ah, CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?

```

```

817 00001020 7409      <1>      jz      short K22B      ; GO IF NOT
818                      <1>      ;
819 00001022 6650      <1>      push     ax      ; SAVE SCAN CODE AND SHIFT MASK
820 00001024 E8F9030000 <1>      call     SND_LED      ; GO TURN MODE INDICATORS ON
821 00001029 6658      <1>      pop      ax      ; RESTORE SCAN CODE
822                      <1> K22B:
823 0000102B 3C52      <1>      cmp     al, INS_KEY      ; TEST FOR 1ST MAKE OF INSERT KEY
824 0000102D 0F8583000000 <1>      jne     K26      ; JUMP IF NOT INSERT KEY
825 00001033 88C4      <1>      mov     ah, al      ; SCAN CODE IN BOTH HALVES OF AX
826 00001035 E999000000 <1>      jmp     K28      ; FLAGS UPDATED, PROC. FOR BUFFER
827                      <1>      ;
828                      <1>      ;----- BREAK SHIFT FOUND
829                      <1> K23:      ; BREAK-SHIFT-FOUND
830 0000103A 80FC10      <1>      cmp     ah, SCROLL_SHIFT ; IS THIS A TOGGLE KEY
831 0000103D F6D4      <1>      not     ah      ; INVERT MASK
832 0000103F 7355      <1>      jae     short K24      ; YES, HANDLE BREAK TOGGLE
833 00001041 2025[ED5D0000] <1>      and     [KB_FLAG], ah      ; TURN OFF SHIFT BIT
834 00001047 80FCFB      <1>      cmp     ah, ~CTL_SHIFT      ; IS THIS ALT OR CTL?
835 0000104A 7730      <1>      ja      short K23D      ; NO, ALL DONE
836                      <1>      ;
837 0000104C F6C702      <1>      test    bh, LC_E0      ; 2ND ALT OR CTL?
838 0000104F 7408      <1>      jz      short K23A      ; NO, HANSLE NORMALLY
839 00001051 2025[F05D0000] <1>      and     [KB_FLAG_3], ah      ; RESET BIT FOR RIGHT ALT OR CTL
840 00001057 EB08      <1>      jmp     short K23B      ; CONTINUE
841                      <1> K23A:
842 00001059 D2FC      <1>      sar     ah, cl      ; MOVE THE MASK BIT TWO POSITIONS
843 0000105B 2025[EE5D0000] <1>      and     [KB_FLAG_1], ah      ; RESET BIT FOR LEFT ALT AND CTL
844                      <1> K23B:
845 00001061 88C4      <1>      mov     ah, al      ; SAVE SCAN CODE
846 00001063 A0[F05D0000] <1>      mov     al, [KB_FLAG_3]      ; GET RIGHT ALT & CTRL FLAGS
847 00001068 D2E8      <1>      shr     al, cl      ; MOVE TO BITS 1 & 0
848 0000106A 0A05[EE5D0000] <1>      or      al, [KB_FLAG_1]      ; PUT IN LEFT ALST & CTL FLAGS
849 00001070 D2E0      <1>      shl     al, cl      ; MOVE BACK TO BITS 3 & 2
850 00001072 240C      <1>      and     al, ALT_SHIFT+CTL_SHIFT ; FILTER OUT OTHER GARBAGE
851 00001074 0805[ED5D0000] <1>      or      [KB_FLAG], al      ; PUT RESULT IN THE REAL FLAGS
852 0000107A 88E0      <1>      mov     al, ah
853                      <1> K23D:
854 0000107C 3CB8      <1>      cmp     al, ALT_KEY+80h      ; IS THIS ALTERNATE SHIFT RELEASE
855 0000107E 7536      <1>      jne     short K26      ; INTERRUPT RETURN
856                      <1>      ;
857                      <1>      ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
858 00001080 A0[F15D0000] <1>      mov     al, [ALT_INPUT]
859 00001085 B400      <1>      mov     ah, 0      ; SCAN CODE OF 0
860 00001087 8825[F15D0000] <1>      mov     [ALT_INPUT], ah      ; ZERO OUT THE FIELD
861 0000108D 3C00      <1>      cmp     al, 0      ; WAS THE INPUT = 0?
862 0000108F 7425      <1>      je      short K26      ; INTERRUPT_RETURN
863                      <1>      ; 29/01/2016
864                      <1>      ; jmp     K61      ; IT WASN'T, SO PUT IN BUFFER
865 00001091 E9D0020000 <1>      jmp     _K60
866                      <1>      ;
867                      <1> K24:      ; BREAK-TOGGLE
868 00001096 2025[EE5D0000] <1>      and     [KB_FLAG_1], ah      ; INDICATE NO LONGER DEPRESSED
869 0000109C EB18      <1>      jmp     short K26      ; INTERRUPT_RETURN
870                      <1>      ;
871                      <1>      ;----- TEST FOR HOLD STATE
872                      <1>      ;
873                      <1> K25:      ; NO-SHIFT-FOUND
874 0000109E 3C80      <1>      cmp     al, 80h      ; TEST FOR BREAK KEY
875 000010A0 7314      <1>      jae     short K26      ; NOTHING FOR BREAK CHARS FROM HERE ON
876 000010A2 F605[EE5D0000]08 <1>      test    byte [KB_FLAG_1], HOLD_STATE ; ARE WE IN HOLD STATE
877 000010A9 7428      <1>      jz      short K28      ; BRANCH AROUND TEST IF NOT
878 000010AB 3C45      <1>      cmp     al, NUM_KEY
879 000010AD 7407      <1>      je      short K26      ; CAN'T END HOLD ON NUM_LOCK
880 000010AF 8025[EE5D0000]F7 <1>      and     byte [KB_FLAG_1], ~HOLD_STATE ; TURN OFF THE HOLD STATE BIT
881                      <1>      ;
882                      <1> K26:
883 000010B6 8025[F05D0000]FC <1>      and     byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
884                      <1> K26A:      ; INTERRUPT-RETURN
885 000010BD FA      <1>      cli      ; TURN OFF INTERRUPTS
886 000010BE B020      <1>      mov     al, EOI      ; END OF INTERRUPT COMMAND
887 000010C0 E620      <1>      out     20h, al      ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
888                      <1> K27:      ; INTERRUPT-RETURN-NO-EOI
889 000010C2 B0AE      <1>      mov     al, ENA_KBD      ; INSURE KEYBOARD IS ENABLED
890 000010C4 E8F8020000 <1>      call     SHIP_IT      ; EXECUTE ENABLE
891                      <1> K27A:
892 000010C9 FA      <1>      cli      ; DISABLE INTERRUPTS
893                      <1>      ;mov byte [intflg], 0 ; 07/01/2017 ; 15/01/2017
894 000010CA 07      <1>      pop     es      ; RESTORE REGISTERS
895 000010CB 1F      <1>      pop     ds
896 000010CC 5F      <1>      pop     edi
897 000010CD 5E      <1>      pop     esi
898 000010CE 5A      <1>      pop     edx
899 000010CF 59      <1>      pop     ecx
900 000010D0 5B      <1>      pop     ebx
901 000010D1 58      <1>      pop     eax
902                      <1>      ;pop ebp
903 000010D2 CF      <1>      iretd      ; RETURN
904                      <1>
905                      <1>      ;----- NOT IN HOLD STATE
906                      <1> K28:      ; NO-HOLD-STATE
907 000010D3 3C58      <1>      cmp     al, 88      ; TEST FOR OUT-OF-RANGE SCAN CODES
908 000010D5 77DF      <1>      ja      short K26      ; IGNORE IF OUT-OF-RANGE
909                      <1>      ;
910 000010D7 F6C308      <1>      test    bl, ALT_SHIFT      ; ARE WE IN ALTERNATE SHIFT
911                      <1>      ;jz short K28A      ; IF NOT ALTERNATE
912 000010DA 0F84F1000000 <1>      jz      K38
913                      <1>      ;
914 000010E0 F6C710      <1>      test    bh, KBX      ; IS THIS THE ENCHANCED KEYBOARD?
915 000010E3 740D      <1>      jz      short K29      ; NO, ALT STATE IS REAL
916                      <1>      ;28/02/2015
917 000010E5 F605[EE5D0000]04 <1>      test    byte [KB_FLAG_1], SYS_SHIFT ; YES, IS SYSREQ KEY DOWN?
918                      <1>      ;jz short K29      ; NO, ALT STATE IS REAL

```



```

919 000010EC 0F85DF000000 <1> jnz K38 ; YES, THIS IS PHONY ALT STATE
920 <1> ; <1> ; DUE TO PRESSING SYSREQ
921 <1> ;K28A: jmp short K38
922 <1> ;
923 <1> ;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
924 <1> K29: ; TEST-RESET
925 000010F2 F6C304 <1> test bl, CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO?
926 000010F5 740B <1> jz short K31 ; NO_RESET
927 000010F7 3C53 <1> cmp al, DEL_KEY ; CTL-ALT STATE, TEST FOR DELETE KEY
928 000010F9 7507 <1> jne short K31 ; NO_RESET, IGNORE
929 <1> ;
930 <1> ;----- CTL-ALT-DEL HAS BEEN FOUND
931 <1> ; 26/08/2014
932 <1> cpu_reset:
933 <1> ; IBM PC/AT ROM BIOS source code - 10/06/85 (TEST4.ASM - PROC_SHUTDOWN)
934 <1> ; Send FEH (system reset command) to the keyboard controller.
935 000010FB B0FE <1> mov al, SHUT_CMD ; SHUTDOWN COMMAND
936 000010FD E664 <1> out STATUS_PORT, al ; SEND TO KEYBOARD CONTROL PORT
937 <1> khere:
938 000010FF F4 <1> hlt ; WAIT FOR 80286 RESET
939 00001100 EBFD <1> jmp short khere ; INSURE HALT
940 <1>
941 <1> ;
942 <1> ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
943 <1> K31: ; NO-RESET
944 00001102 3C39 <1> cmp al, 57 ; TEST FOR SPACE KEY
945 00001104 7507 <1> jne short K311 ; NOT THERE
946 00001106 B020 <1> mov al, ' ' ; SET SPACE CHAR
947 00001108 E948020000 <1> jmp K57 ; BUFFER_FILL
948 <1> K311:
949 0000110D 3C0F <1> cmp al, 15 ; TEST FOR TAB KEY
950 0000110F 7509 <1> jne short K312 ; NOT THERE
951 00001111 66B800A5 <1> mov ax, 0A500h ; SET SPECIAL CODE FOR ALT-TAB
952 00001115 E93B020000 <1> jmp K57 ; BUFFER_FILL
953 <1> K312:
954 0000111A 3C4A <1> cmp al, 74 ; TEST FOR KEY PAD -
955 0000111C 0F84A2000000 <1> je K37B ; GO PROCESS
956 00001122 3C4E <1> cmp al, 78 ; TEST FOR KEY PAD +
957 00001124 0F849A000000 <1> je K37B ; GO PROCESS
958 <1> ;
959 <1> ;----- LOOK FOR KEY PAD ENTRY
960 <1> K32: ; ALT-KEY-PAD
961 0000112A BF[B05C0000] <1> mov edi, K30 ; ALT-INPUT-TABLE offset
962 0000112F B90A000000 <1> mov ecx, 10 ; LOOK FOR ENTRY USING KEYPAD
963 00001134 F2AE <1> repne scasb ; LOOK FOR MATCH
964 00001136 7525 <1> jne short K33 ; NO_ALT_KEYPAD
965 00001138 F6C702 <1> test bh, LC_E0 ; IS THIS ONE OF THE NEW KEYS?
966 0000113B 0F858A000000 <1> jnz K37C ; YES, JUMP, NOT NUMPAD KEY
967 00001141 81EF[B15C0000] <1> sub edi, K30+1 ; DI NOW HAS ENTRY VALUE
968 00001147 A0[F15D0000] <1> mov al, [ALT_INPUT] ; GET THE CURRENT BYTE
969 0000114C B40A <1> mov ah, 10 ; MULTIPLY BY 10
970 0000114E F6E4 <1> mul ah
971 00001150 6601F8 <1> add ax, di ; ADD IN THE LATEST ENTRY
972 00001153 A2[F15D0000] <1> mov [ALT_INPUT], al ; STORE IT AWAY
973 <1> ;K32A:
974 00001158 E959FFFFFF <1> jmp K26 ; THROW AWAY THAT KEYSTROKE
975 <1> ;
976 <1> ;----- LOOK FOR SUPERSHIFT ENTRY
977 <1> K33: ; NO-ALT-KEYPAD
978 0000115D C605[F15D0000]00 <1> mov byte [ALT_INPUT], 0 ; ZERO ANY PREVIOUS ENTRY INTO INPUT
979 00001164 B91A000000 <1> mov ecx, 26 ; (DI),(ES) ALREADY POINTING
980 00001169 F2AE <1> repne scasb ; LOOK FOR MATCH IN ALPHABET
981 0000116B 7450 <1> je short K37A ; MATCH FOUND, GO FILL THE BUFFER
982 <1> ;
983 <1> ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
984 <1> K34: ; ALT-TOP-ROW
985 0000116D 3C02 <1> cmp al, 2 ; KEY WITH '1' ON IT
986 0000116F 7253 <1> jnb short K37B ; MUST BE ESCAPE
987 00001171 3C0D <1> cmp al, 13 ; IS IT IN THE REGION
988 00001173 7705 <1> ja short K35 ; NO, ALT SOMETHING ELSE
989 00001175 80C476 <1> add ah, 118 ; CONVERT PSEUDO SCAN CODE TO RANGE
990 00001178 EB43 <1> jmp short K37A ; GO FILL THE BUFFER
991 <1> ;
992 <1> ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
993 <1> K35: ; ALT-FUNCTION
994 0000117A 3C57 <1> cmp al, F11_M ; IS IT F11?
995 0000117C 7209 <1> jnb short K35A ; 20/02/2015 ; NO, BRANCH
996 0000117E 3C58 <1> cmp al, F12_M ; IS IT F12?
997 00001180 7705 <1> ja short K35A ; 20/02/2015 ; NO, BRANCH
998 00001182 80C434 <1> add ah, 52 ; CONVERT TO PSEUDO SCAN CODE
999 00001185 EB36 <1> jmp short K37A ; GO FILL THE BUFFER
1000 <1> K35A:
1001 00001187 F6C702 <1> test bh, LC_E0 ; DO WE HAVE ONE OF THE NEW KEYS?
1002 0000118A 7422 <1> jz short K37 ; NO, JUMP
1003 0000118C 3C1C <1> cmp al, 28 ; TEST FOR KEYPAD ENTER
1004 0000118E 7509 <1> jne short K35B ; NOT THERE
1005 00001190 66B800A6 <1> mov ax, 0A600h ; SPECIAL CODE
1006 00001194 E9BC010000 <1> jmp K57 ; BUFFER FILL
1007 <1> K35B:
1008 00001199 3C53 <1> cmp al, 83 ; TEST FOR DELETE KEY
1009 0000119B 742E <1> je short K37C ; HANDLE WITH OTHER EDIT KEYS
1010 0000119D 3C35 <1> cmp al, 53 ; TEST FOR KEYPAD /
1011 <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> jnb 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

```


[illegible]

```

1123
1124 0000127A BB[E45C0000]
1125 0000127F 0F82AE000000
1126 00001285 E9B9000000
1127
1128
1129
1130 0000128A 3C37
1131 0000128C 7528
1132 0000128E F6C710
1133 00001291 7407
1134 00001293 F6C702
1135 00001296 7507
1136 00001298 EB41
1137
1138 0000129A F6C303
1139 0000129D 743C
1140
1141
1142
1143 0000129F B0AE
1144 000012A1 E81B010000
1145 000012A6 B020
1146 000012A8 E620
1147
1148
1149
1150
1151 000012AA 8025[F05D0000]FC
1152 000012B1 E90CFEFFFF
1153
1154
1155
1156 000012B6 3C3A
1157 000012B8 7734
1158 000012BA 3C35
1159 000012BC 7505
1160 000012BE F6C702
1161 000012C1 7518
1162
1163 000012C3 B91A000000
1164 000012C8 BF[BA5C0000]
1165 000012CD F2AE
1166
1167 000012CF 7505
1168
1169 000012D1 F6C340
1170 000012D4 750C
1171
1172 000012D6 F6C303
1173 000012D9 750C
1174
1175
1176 000012DB BB[3C5D0000]
1177 000012E0 EB51
1178
1179 000012E2 F6C303
1180 000012E5 75F4
1181
1182 000012E7 BB[945D0000]
1183 000012EC EB45
1184
1185
1186
1187 000012EE 3C44
1188
1189
1190 000012F0 7635
1191
1192
1193
1194 000012F2 3C53
1195 000012F4 772D
1196
1197
1198
1199 000012F6 3C4A
1200 000012F8 74ED
1201 000012FA 3C4E
1202 000012FC 74E9
1203 000012FE F6C702
1204 00001301 750A
1205
1206 00001303 F6C320
1207 00001306 7514
1208 00001308 F6C303
1209
1210 0000130B 75DA
1211
1212
1213
1214 0000130D 3C4C
1215 0000130F 7504
1216 00001311 B0F0
1217 00001313 EB40
1218
1219 00001315 BB[3C5D0000]
1220 0000131A EB27
1221
1222
1223
1224 0000131C F6C303

<1> K42B:
<1>      mov     ebx, _K8           ; SET UP TO TRANSLATE CTL
<1>      jb      K56 ; 20/02/2015
<1>      jmp     K64
<1>      ;
<1>      ;----- NOT IN CONTROL SHIFT
<1> K44:
<1>      cmp     al, 55             ; NOT-CTL-SHIFT
<1>      jne     short K45         ; PRINT SCREEN KEY?
<1>      test    bh, KBX           ; IS THIS ENHANCED KEYBOARD?
<1>      jz      short K44A        ; NO, TEST FOR SHIFT STATE
<1>      test    bh, LC_E0         ; YES, LAST CODE A MARKER?
<1>      jnz     short K44B        ; YES, IS PRINT SCREEN
<1>      jmp     short K45C        ; NO, TRANSLATE TO '*' CHARACTER
<1> K44A:
<1>      test    bl, LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
<1>      jz      short K45C        ; NO, TRANSLATE TO '*' CHARACTER
<1>      ;
<1>      ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
<1> K44B:
<1>      mov     al, ENA_KBD       ; INSURE KEYBOARD IS ENABLED
<1>      call    SHIP_IT           ; EXECUTE ENABLE
<1>      mov     al, EOI           ; END OF CURRENT INTERRUPT
<1>      out     20h, al ;out INTA00, al ; SO FURTHER THINGS CAN HAPPEN
<1>      ; Print Screen !!!       ; ISSUE PRINT SCREEN INTERRUPT (INT 05h)
<1>      ;PUSH BP                 ; SAVE POINTER
<1>      ;INT 5H                  ; ISSUE PRINT SCREEN INTERRUPT
<1>      ;POP BP                  ; RESTORE POINTER
<1>      and     byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; ZERO OUT THESE FLAGS
<1>      jmp     K27               ; GO BACK WITHOUT EOI OCCURRING
<1>      ;
<1>      ;----- HANDLE IN-CORE KEYS
<1> K45:
<1>      cmp     al, 58             ; NOT-PRINT-SCREEN
<1>      ja      short K46         ; TEST FOR IN-CORE AREA
<1>      cmp     al, 53             ; JUMP IF NOT
<1>      jne     short K45A        ; IS THIS THE '/' KEY?
<1>      test    bh, LC_E0         ; NO, JUMP
<1>      jnz     short K45C        ; WAS THE LAST CODE THE MARKER?
<1>      ; YES, TRANSLATE TO CHARACTER
<1> K45A:
<1>      mov     ecx, 26            ; LENGHT OF SEARCH
<1>      mov     edi, K30+10       ; POINT TO TABLE OF A-Z CHARS
<1>      repne   scasb             ; IS THIS A LETTER KEY?
<1>      ; 20/02/2015
<1>      jne     short K45B        ; NO, SYMBOL KEY
<1>      ;
<1>      test    bl, CAPS_STATE     ; ARE WE IN CAPS_LOCK?
<1>      jnz     short K45D        ; TEST FOR SURE
<1> K45B:
<1>      test    bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
<1>      jnz     short K45E        ; YES, UPPERCASE
<1>      ; NO, LOWERCASE
<1> K45C:
<1>      mov     ebx, K10          ; TRANSLATE TO LOWERCASE LETTERS
<1>      jmp     short K56
<1> K45D:
<1>      test    bl, LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
<1>      jnz     short K45C        ; SHIFTED TEMP OUT OF CAPS STATE
<1> K45E:
<1>      mov     ebx, K11          ; TRANSLATE TO UPPER CASE LETTERS
<1> K45F:
<1>      jmp     short K56
<1>      ;
<1>      ;----- TEST FOR KEYS F1 - F10
<1> K46:
<1>      cmp     al, 68             ; NOT IN-CORE AREA
<1>      ja      short K47         ; TEST FOR F1 - F10
<1>      jmp     short K53         ; JUMP IF NOT
<1>      jna     short K53         ; YES, GO DO FN KEY PROCESS
<1>      ;
<1>      ;----- HANDLE THE NUMERIC PAD KEYS
<1> K47:
<1>      cmp     al, 83             ; NOT F1 - F10
<1>      ja      short K52         ; TEST NUMPAD KEYS
<1>      ; JUMP IF NOT
<1>      ;
<1>      ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
<1> K48:
<1>      cmp     al, 74             ; SPECIAL CASE FOR MINUS
<1>      je      short K45E        ; GO TRANSLATE
<1>      cmp     al, 78             ; SPECIAL CASE FOR PLUS
<1>      je      short K45E        ; GO TRANSLATE
<1>      test    bh, LC_E0         ; IS THIS ONE OF THE NEW KEYS?
<1>      jnz     short K49         ; YES, TRANSLATE TO BASE STATE
<1>      ;
<1>      test    bl, NUM_STATE     ; ARE WE IN NUM LOCK
<1>      jnz     short K50         ; TEST FOR SURE
<1>      test    bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
<1>      jnz     short K51         ; IF SHIFTED, REALLY NUM STATE
<1>      jnz     short K45E
<1>      ;
<1>      ;----- BASE CASE FOR KEYPAD
<1> K49:
<1>      cmp     al, 76             ; SPECIAL CASE FOR BASE STATE 5
<1>      jne     short K49A        ; CONTINUE IF NOT KEYPAD 5
<1>      mov     al, 0F0h          ; SPECIAL ASCII CODE
<1>      jmp     short K57         ; BUFFER FILL
<1> K49A:
<1>      mov     ebx, K10          ; BASE CASE TABLE
<1>      jmp     short K64         ; CONVERT TO PSEUDO SCAN
<1>      ;
<1>      ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
<1> K50:
<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[945D0000] <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[F05D0000]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[F05D0000]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>      jnb      short K61
1276 0000136B 80FC6F     <1>      cmp      ah, 6Fh          ; ALT + F8 key
1277 0000136E 771A      <1>      ja       short K61
1278                   <1>      ;
1279 00001370 8A1D[4E380100] <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[FE5D0000] <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[FA5D0000] <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[FE5D0000] <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> SD0:
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[EF5D0000]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[EF5D0000]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[EF5D0000]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[EF5D0000]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[EF5D0000]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[EF5D0000]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[EF5D0000]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[EF5D0000]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[EF5D0000]F8 <1>      and     byte [KB_FLAG_2], 0F8h ; ~KB_LEDS ; CLEAR MODE INDICATOR BITS
1414 0000145E 0805[EF5D0000] <1>      or      [KB_FLAG_2], al        ; SAVE PRESENT INDICATORS FOR NEXT TIME
1415 00001464 F605[EF5D0000]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[EF5D0000]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[EF5D0000]3F <1>      and     byte [KB_FLAG_2], ~(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
1428                      <1> SL1:

```



```

1429 0000148B FB      <1>      sti                      ; ENABLE INTERRUPTS
1430 0000148C C3      <1>      retn                     ; RETURN TO CALLER
1431
1432
1433      <1> MAKE_LED:
1434      <1>      ;-----
1435      <1>      ; MAKE_LED
1436      <1>      ;      THIS ROUTINES FORMS THE DATA BYTE NECESSARY TO TURN ON/OFF
1437      <1>      ;      THE MODE INDICATORS.
1438      <1>      ;-----
1439      <1>      ;
1440 0000148D A0[ED5D0000] <1>      ;push  cx          ; SAVE CX
1441 00001492 2470      <1>      mov  al, [KB_FLAG]      ; GET CAPS & NUM LOCK INDICATORS
1442      <1>      and  al, CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
1443      <1>      ;mov  cl, 4          ; SHIFT COUNT
1444 00001494 C0C004      <1>      ;rol  al, cl          ; SHIFT BITS OVER TO TURN ON INDICATORS
1445 00001497 2407      <1>      rol  al, 4 ; 20/02/2015
1446      <1>      and  al, 07h          ; MAKE SURE ONLY MODE BITS ON
1447 00001499 C3      <1>      ;pop  cx
1448      <1>      retn                     ; RETURN TO CALLER
1449
1450      <1> ; % include 'kybdata.s'      ; KEYBOARD DATA
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)= 0DH      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 TO 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      :
170     <1> ;      (AL)= 01H      WRITE CHARACTER STRING AND MOVE CURSOR      :
171     <1> ;      BL - ATTRIBUTE      :
172     <1> ;      STRING IS <CHAR,CHAR, ... ,CHAR>      :
173     <1> ;      CURSOR MOVED      :
174     <1> ;      (AL)= 02H      WRITE CHARACTER AND ATTRIBUTE STRING      :
175     <1> ;      (VALID FOR ALPHA MODES ONLY)      :
176     <1> ;      STRING IS <CHAR,ATTR,CHAR,ATTR ... ,CHAR,ATTR>      :
177     <1> ;      CURSOR IS NOT MOVED      :
178     <1> ;      (AL)= 03H WRITE CHARACTER AND ATTRIBUTE STRING AND MOVE CURSOR :

```

```
179 <1> ; (VALID FOR ALPHA MODES ONLY) :
180 <1> ; STRING IS <CHAR,ATTR,CHAR,ATTR .. ,CHAR,ATTR> :
181 <1> ; CURSOR IS MOVED :
182 <1> ; NOTE: CARRIAGE RETURN, LINE FEED, BACKSPACE, AND BELL ARE :
183 <1> ; TREATED AS COMMANDS RATHER THAN PRINTABLE CHARACTERS. :
184 <1> ; :
185 <1> ; BX,CX,DX,SI,DI,BP,SP,DS,ES,SS PRESERVED DURING CALLS EXCEPT FOR :
186 <1> ; BX,CX,DX RETURN VALUES ON FUNCTIONS 03H,04H,0DH AND 0FH. ON ALL CALLS :
187 <1> ; AX IS MODIFIED. :
188 <1> ;-----
189 <1>
190 000014A2 [4F150000] <1> M1: dd SET_MODE ; TABLE OF ROUTINES WITHIN VIDEO I/O
191 000014A6 [B7180000] <1> dd SET_CTYPE
192 000014AA [EB180000] <1> dd SET_CPOS
193 000014AE [13190000] <1> dd READ_CURSOR
194 <1> ;dd VIDEO_RETURN ; READ_LPEN
195 000014B2 [38150000] <1> dd set_mode_ncm ; Set mode without clearing video memory
196 000014B6 [59190000] <1> dd ACT_DISP_PAGE
197 000014BA [F0190000] <1> dd SCROLL_UP
198 000014BE [141B0000] <1> dd SCROLL_DOWN
199 000014C2 [951B0000] <1> dd READ_AC_CURRENT
200 000014C6 [ED1B0000] <1> dd WRITE_AC_CURRENT
201 000014CA [131C0000] <1> dd WRITE_C_CURRENT
202 000014CE [39250000] <1> dd SET_COLOR
203 000014D2 [A4250000] <1> dd WRITE_DOT
204 000014D6 [6F250000] <1> dd READ_DOT
205 000014DA [951C0000] <1> dd WRITE_TTY
206 000014DE [20150000] <1> dd VIDEO_STATE
207 000014E2 [EF2E0000] <1> dd vga_pal_funcs ; 10/08/2016 (TRDOS 386)
208 000014E6 [A52A0000] <1> dd font_setup ; 10/07/2016 (TRDOS 386)
209 000014EA [54150000] <1> dd VIDEO_RETURN ; RESERVED
210 000014EE [021E0000] <1> dd WRITE_STRING ; 23/06/2016 (TRDOS 386)
211 <1> M1L EQU $ - M1
212 <1>
213 <1> ; 14/01/2017
214 <1> ; 02/01/2017
215 <1> ; 04/07/2016
216 <1> ; 12/05/2016
217 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
218 <1> int31h: ; Video BIOS
219 <1>
220 <1> ; BH = Video page number
221 <1> ; BL = Color/Attribute
222 <1> ; AH = Function number
223 <1> ; AL = Character
224 <1>
225 <1> VIDEO_IO_1:
226 <1> ;sti ; INTERRUPTS BACK ON
227 000014F2 FC <1> cld ; SET DIRECTION FORWARD
228 000014F3 80FC14 <1> cmp ah, M1L/4 ; TEST FOR WITHIN TABLE RANGE
229 000014F6 7327 <1> jnb short M4 ; BRANCH TO EXIT IF NOT A VALID COMMAND
230 <1>
231 000014F8 06 <1> push es
232 000014F9 1E <1> push ds ; SAVE WORK AND PARAMETER REGISTERS
233 000014FA 52 <1> push edx
234 000014FB 51 <1> push ecx
235 000014FC 53 <1> push ebx
236 000014FD 56 <1> push esi
237 000014FE 57 <1> push edi
238 000014FF 55 <1> push ebp
239 <1>
240 00001500 66BE1000 <1> mov si, KDATA ; POINT DS: TO DATA SEGMENT
241 00001504 8EDE <1> mov ds, si
242 00001506 8EC6 <1> mov es, si
243 00001508 BF00800B00 <1> mov edi, 0B8000h ; GET offset FOR COLOR CARD
244 0000150D A3[AC450100] <1> mov [video_eax], eax ; 12/05/2016
245 <1> ; 23/03/2016
246 00001512 C0E402 <1> shl ah, 2 ; dword ; TIMES 2 FOR WORD TABLE LOOKUP
247 00001515 0FB6F4 <1> movzx esi, ah ; MOVE OFFSET INTO LOOK UP REGISTER (SI)
248 <1> ;mov ah, [CRT_MODE] ; MOVE CURRENT MODE INTO (AH) REGISTER
249 <1>
250 <1> ;;15/01/2017
251 <1> ; 14/01/2017
252 <1> ; 02/01/2017
253 <1> ;mov byte [intflg], 31h ; video interrupt
254 00001518 FB <1> sti
255 <1> ;
256 <1>
257 00001519 FFA6[A2140000] <1> JMP dword [esi+M1] ; GO TO SELECTED FUNCTION
258 <1>
259 <1> M4: ; COMMAND NOT VALID
260 0000151F CF <1> iretd ; DO NOTHING IF NOT IN VALID RANGE
261 <1>
262 <1> VIDEO_STATE:
263 <1> ; 26/06/2016
264 <1> ; 12/05/2016
265 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
266 <1>
267 <1> ;-----
268 <1> ; VIDEO STATE
269 <1> ; RETURNS THE CURRENT VIDEO STATE IN AX
270 <1> ; AH = NUMBER OF COLUMNS ON THE SCREEN
271 <1> ; AL = CURRENT VIDEO MODE
272 <1> ; BH = CURRENT ACTIVE PAGE
273 <1> ;-----
274 <1>
275 00001520 8A25[245E0000] <1> mov ah, [CRT_COLS] ; GET NUMBER OF COLUMNS
276 00001526 A0[225E0000] <1> mov al, [CRT_MODE] ; CURRENT MODE
277 <1> ;movzx esi, al
278 <1> ;mov ah, [esi+M6]
279 <1> ; BH = active page
280 0000152B 8A3D[4E380100] <1> mov bh, [ACTIVE_PAGE] ; GET CURRENT ACTIVE PAGE
```

```

281 00001531 FA      <1>      cli      ; 02/01/2017
282 00001532 5D      <1>      pop      ebp      ; RECOVER REGISTERS
283 00001533 5F      <1>      pop      edi
284 00001534 5E      <1>      pop      esi
285 00001535 59      <1>      pop      ecx      ; DISCARD SAVED BX
286 00001536 EB26    <1>      jmp      short M15      ; RETURN TO CALLER
287
288 <1> set_mode_ncm:
289 <1>      ; 04/07/2016 - TRDOS 386 (TRDOS v2.0)
290 <1>      ; set mode without clearing the video memory
291 <1>      ; (only for graphics modes)
292 00001538 3C07    <1>      cmp      al, 7 ; IBM PC CGA modes
293 0000153A 7613    <1>      jna      short SET_MODE ; normal function (clear)
294 <1>      ; do not clear memory
295 0000153C A2[BB450100] <1>      mov      [noclearmem], al ; > 0
296 00001541 E81F000000 <1>      call     _set_mode
297 00001546 C605[BB450100]00 <1>      mov      byte [noclearmem], 0
298 0000154D EB05    <1>      jmp      short VIDEO_RETURN
299
300 <1>      ; 10/08/2016
301 <1>      ; 08/08/2016
302 <1>      ; 30/07/2016
303 <1>      ; 29/07/2016
304 <1>      ; 27/07/2016
305 <1>      ; 26/07/2016
306 <1>      ; 25/07/2016
307 <1>      ; 23/07/2016
308 <1>      ; 18/07/2016
309 <1>      ; 02/07/2016
310 <1>      ; 26/06/2016
311 <1>      ; 24/06/2016
312 <1>      ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
313 <1> SET_MODE:
314 <1>      ; For 32 bit TRDOS and Retro UNIX 386:
315 <1>      ;      valid video mode: 03h only!
316 <1>      ;      (VGA modes will be selected with another routine)
317 <1>      ;
318 <1>      ; set_txt_mode ; 80*25 (16 fore colors, 8 back colors)
319 <1>
320 <1> ;-----
321 <1> ; SET MODE :
322 <1> ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO :
323 <1> ; THE SELECTED MODE, THE SCREEN IS BLANKED. :
324 <1> ; INPUT :
325 <1> ; (AL) - MODE SELECTED (RANGE 0-7) :
326 <1> ; OUTPUT :
327 <1> ; NONE :
328 <1> ;-----
329 <1>
330 0000154F E811000000 <1>      call     _set_mode ; 24/06/2016 (set_txt_mode)
331 <1>
332 <1> ; 12/05/2016
333 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
334 <1>
335 <1> ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
336 <1>
337 <1> VIDEO_RETURN:
338 00001554 A1[AC450100] <1>      mov      eax, [video_eax] ; 12/05/2016
339 <1> _video_return:
340 00001559 FA      <1>      cli ; 02/01/2017
341 0000155A 5D      <1>      pop      ebp
342 0000155B 5F      <1>      pop      edi
343 0000155C 5E      <1>      pop      esi
344 0000155D 5B      <1>      pop      ebx
345 <1> M15: ; VIDEO_RETURN_C
346 <1>      ; ;15/01/2017
347 <1>      ; 02/01/2017
348 <1>      ; ;mov byte [intflg], 0
349 <1>      ;
350 0000155E 59      <1>      pop      ecx
351 0000155F 5A      <1>      pop      edx
352 00001560 1F      <1>      pop      ds
353 00001561 07      <1>      pop      es ; RECOVER SEGMENTS
354 00001562 CF      <1>      iretd ; ALL DONE
355 <1>
356 <1> set_txt_mode:
357 <1>      ; 29/07/2016
358 <1>      ; 27/06/2016
359 00001563 B003    <1>      mov      al, 3
360 <1>
361 <1> ; 10/08/2016
362 <1> ; 08/08/2016
363 <1> ; 30/07/2016
364 <1> ; 29/07/2016
365 <1> ; 27/07/2016
366 <1> ; 26/07/2016
367 <1> ; 25/07/2016
368 <1> ; 23/07/2016
369 <1> ; 18/07/2016
370 <1> ; 07/07/2016
371 <1> ; 04/07/2016
372 <1> ; 03/07/2016
373 <1> ; 02/07/2016
374 <1> ; 26/06/2016
375 <1> ; 24/06/2016 (set_txt_mode -> _set_mode)
376 <1> ; 17/06/2016
377 <1> ; 29/05/2016
378 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
379 <1> _set_mode:
380 <1>      ; 24/06/2016
381 00001565 3805[225E0000] <1>      cmp      [CRT_MODE], al ; current mode = requested mode ?
382 0000156B 750D    <1>      jne      short _sm_0

```

```

383 0000156D 3C03      <1>      cmp     al, 3          ; text, 80*25 color, default mode
384                    <1>                        ; for TRDOS 386 MainProg
385 0000156F 755F      <1>      jne     short _sm_2      ; multiscreen is only for mode 3
386                    <1>
387                    <1>      ; If '_set_mode' procedure is called for video mode 3
388                    <1>      ;       while video mode is 3, video page will be cleared
389                    <1>      ;       and cursor position of video page will be reset.
390                    <1>
391                    <1>      ; 29/07/2016
392 00001571 800D[B9450100]80 <1>      or      byte [p_crt_model], 80h ; clear page indicator
393 00001578 EB5B      <1>      jmp     short _sm_3
394                    <1> _sm_0:
395 0000157A 803D[225E0000]03 <1>      cmp     byte [CRT_MODE], 3
396 00001581 7534      <1>      jne     short _sm_1
397                    <1>
398                    <1>      ; If '_set_mode' procedure is called for a video mode
399                    <1>      ;       except video mode 3, while current video mode
400                    <1>      ;       is 3; all video pages of mode 3 will be copied
401                    <1>      ;       to 98000h address as backup, before mode change.
402                    <1>
403                    <1> _sm_save_pm:
404                    <1>      ; 03/07/2016
405                    <1>      ; save video pages
406 00001583 BE00800B00 <1>      mov     esi, 0B8000h
407 00001588 BF00800900 <1>      mov     edi, 98000h ; 30/07/2016
408 0000158D B900200000 <1>      mov     ecx, (0B8000h-0B0000h)/4
409 00001592 F3A5      <1>      rep     movsd
410                    <1>
411 00001594 C605[B9450100]03 <1>      mov     byte [p_crt_model], 3 ; previous mode, backup sign
412                    <1>      ;mov     cl, [ACTIVE_PAGE]
413                    <1>      ;mov     [p_crt_page], cl
414                    <1>
415                    <1>      ; save cursor positions
416 0000159B BE[3E380100] <1>      mov     esi, CURSOR_POSN
417 000015A0 BF[BE450100] <1>      mov     edi, cursor_pposn ; cursor positions backup
418 000015A5 B104      <1>      mov     cl, 4
419 000015A7 F3A5      <1>      rep     movsd
420                    <1>
421                    <1>      ; 29/07/2016
422                    <1>      ;mov     [ACTIVE_PAGE], cl ; 0
423 000015A9 860D[4E380100] <1>      xchg     cl, [ACTIVE_PAGE]
424 000015AF 880D[BA450100] <1>      mov     [p_crt_page], cl ; previous page (for mode 3)
425                    <1>      ; [ACTIVE_PAGE] = 0
426 000015B5 EB19      <1>      jmp     short _sm_2
427                    <1>
428                    <1> _sm_1:
429 000015B7 3C03      <1>      cmp     al, 3          ; text, 80*25 color, default mode
430                    <1>                        ; for TRDOS 386 MainProg
431 000015B9 7515      <1>      jne     short _sm_2      ; multiscreen is only for mode 3
432                    <1>
433                    <1>      ; If '_set_mode' procedure is called for video mode 3
434                    <1>      ;       while video mode is not 3 and if there is video
435                    <1>      ;       page backup for video mode 3, all (of 8) mode 3
436                    <1>      ;       video pages will be restored from 98000h.
437                    <1>
438 000015BB 803D[B9450100]03 <1>      cmp     byte [p_crt_model], 3 ; previous mode, backup sign
439 000015C2 750C      <1>      jne     short _sm_2 ; there is no (multiscreen) video pages
440                    <1>                        ; to be restored
441 000015C4 8A0D[BA450100] <1>      mov     cl, [p_crt_page]
442 000015CA 880D[4E380100] <1>      mov     [ACTIVE_PAGE], cl
443                    <1>
444                    <1> _sm_2:
445 000015D0 A2[225E0000] <1>      mov     [CRT_MODE], al ; save mode in global variable
446                    <1> _sm_3:
447                    <1>      ; 30/07/2016
448                    <1>      ; 26/07/2016
449                    <1>      ; 25/07/2016
450                    <1>      ; set_mode_vga:
451                    <1>      ; 18/07/2016
452                    <1>      ; 14/07/2016
453                    <1>      ; 09/07/2016
454                    <1>      ; 04/07/2016
455                    <1>      ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
456                    <1>      ; /// video mode 13h ///
457                    <1>      ; derived from 'Plex86/Bochs VGABios' source code
458                    <1>      ; vgabios-0.7a (2011)
459                    <1>      ; by the LGPL VGABios developers Team (2001-2008)
460                    <1>      ; 'vgabios.c', 'vgatables.h'
461                    <1>      ;
462                    <1>      ; Oracle VirtualBox 5.0.24 VGABios Source Code
463                    <1>      ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
464                    <1>      ;
465 000015D5 88C4      <1>      mov     ah, al
466 000015D7 B910000000 <1>      mov     ecx, vga_mode_count
467 000015DC BE[3E5E0000] <1>      mov     esi, vga_modes
468 000015E1 31DB      <1>      xor     ebx, ebx
469                    <1> _sm_4:
470 000015E3 AC        <1>      lodsb
471 000015E4 38C4      <1>      cmp     ah, al
472 000015E6 740C      <1>      je      short _sm_5
473 000015E8 FEC3      <1>      inc     bl
474 000015EA E2F7      <1>      loop    _sm_4
475                    <1>
476                    <1>      ; UNIMPLEMENTED VIDEO MODE !
477 000015EC 31C0      <1>      xor     eax, eax
478 000015EE A3[AC450100] <1>      mov     [video_eax], eax ; 0
479 000015F3 C3        <1>      retn
480                    <1>
481                    <1> ;----- eBX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
482                    <1>
483                    <1> _sm_5:
484 000015F4 89DE      <1>      mov     esi, ebx

```

```

485 000015F6 81C6[8E5E0000] <1> add esi, vga_memmodel
486 000015FC 8A06 <1> mov al, [esi]
487 000015FE A2[D2450100] <1> mov [VGA_MTYPE], al
488 <1>
489 00001603 89DF <1> mov edi, ebx
490 00001605 81C7[9E5E0000] <1> add edi, vga_dac_s
491 0000160B C0E302 <1> shl bl, 2 ; byte -> dword
492 0000160E 81C3[4E5E0000] <1> add ebx, vga_mode_tbl_ptr
493 <1>
494 <1> ;mov dword [VGA_BASE], 0B8000h
495 <1> ;cmp ah, 0Dh ; [CRT_MODE]
496 <1> ;jb short M9
497 <1> ;mov dword [VGA_BASE], 0A0000h
498 <1> ;M9:
499 00001614 8B33 <1> mov esi, [ebx]
500 00001616 89F3 <1> mov ebx, esi
501 00001618 83C614 <1> add esi, vga_p_cm_pos ; ebx + 20
502 0000161B 668B06 <1> mov ax, [esi] ; get the cursor mode from the table
503 0000161E 66A3[3B5E0000] <1> mov [CURSOR_MODE], ax ; save cursor mode (initial value)
504 <1> ; al = 6, ah = 7
505 <1> ; al = 0Dh, ah = 0Eh ; 25/07/2016
506 00001624 E83B020000 <1> call cursor_shape_fix
507 <1> ; al = 14, ah = 15 (If [CHAR_HEIGHT] = 16)
508 00001629 668906 <1> mov [esi], ax
509 <1>
510 0000162C 56 <1> push esi ; *
511 <1>
512 0000162D 8A25[295E0000] <1> mov ah, [VGA_MODESET_CTL]
513 00001633 80E408 <1> and ah, 8 ; default palette loading ?
514 00001636 7524 <1> jnz short _sm_6
515 00001638 66BAC603 <1> mov dx, 3C6h ; VGAREG_PEL_MASK (DAC mask register)
516 0000163C B0FF <1> mov al, 0FFh ; PEL mask
517 0000163E EE <1> out dx, al
518 0000163F 8A27 <1> mov ah, [edi] ; DAC model (selection number)
519 00001641 E8ED0F0000 <1> call load_dac_palette
520 <1> ; ecx = 0
521 00001646 F605[295E0000]02 <1> test byte [VGA_MODESET_CTL], 2 ; gray scale summing
522 0000164D 740D <1> jz short _sm_6
523 0000164F 53 <1> push ebx
524 00001650 29DB <1> sub ebx, ebx ; sub bl, bl
525 00001652 66B90001 <1> mov cx, 256
526 00001656 E82B100000 <1> call gray_scale_summing
527 0000165B 5B <1> pop ebx
528 <1> _sm_6:
529 <1> ; Reset Attribute Ctl flip-flop
530 0000165C 66BADA03 <1> mov dx, 3DAh ; VGAREG_ACTL_RESET
531 00001660 EC <1> in al, dx
532 <1> ; Set Attribute Ctl
533 00001661 89DE <1> mov esi, ebx ; addr of params tbl for selected mode
534 00001663 83C623 <1> add esi, 35 ; actl regs
535 00001666 30E4 <1> xor ah, ah ; 0
536 00001668 66BAC003 <1> mov dx, 3C0h ; VGAREG_ACTL_ADDRESS
537 <1> _sm_7:
538 0000166C 88E0 <1> mov al, ah
539 0000166E EE <1> out dx, al ; index
540 0000166F AC <1> lodsb
541 <1> ; DX = 3C0h = VGAREG_ACTL_WRITE_DATA
542 00001670 EE <1> out dx, al ; value
543 00001671 FEC4 <1> inc ah
544 00001673 80FC14 <1> cmp ah, 20 ; number of actl registers
545 00001676 72F4 <1> jb short _sm_7
546 <1> ;
547 00001678 88E0 <1> mov al, ah ; 20
548 0000167A EE <1> out dx, al ; index
549 0000167B 28C0 <1> sub al, al ; 0
550 0000167D EE <1> out dx, al ; value
551 <1> ;
552 <1> ; Set Sequencer Ctl
553 0000167E 89DE <1> mov esi, ebx ; addr of params tbl for selected mode
554 00001680 83C605 <1> add esi, 5 ; sequ regs
555 <1> ;
556 00001683 66BAC403 <1> mov dx, 3C4h ; VGAREG_SEQU_ADDRESS
557 00001687 EE <1> out dx, al ; 0
558 00001688 6642 <1> inc dx ; 3C5h ; VGAREG_SEQU_DATA
559 0000168A B003 <1> mov al, 3
560 0000168C EE <1> out dx, al
561 0000168D B401 <1> mov ah, 1
562 <1> _sm_8:
563 0000168F 88E0 <1> mov al, ah
564 <1> ;mov dx, 3C4h ; VGAREG_SEQU_ADDRESS
565 00001691 664A <1> dec dx
566 00001693 EE <1> out dx, al ; index
567 00001694 AC <1> lodsb
568 00001695 6642 <1> inc dx ; 3C5h ; VGAREG_SEQU_DATA
569 00001697 EE <1> out dx, al
570 00001698 80FC04 <1> cmp ah, 4 ; number of sequ regs
571 0000169B 7304 <1> jnb short _sm_9
572 0000169D FEC4 <1> inc ah
573 0000169F EBEE <1> jmp short _sm_8
574 <1> _sm_9:
575 <1> ; Set GrafX Ctl
576 000016A1 89DE <1> mov esi, ebx ; addr of params tbl for selected mode
577 000016A3 83C637 <1> add esi, 55 ; grdc regs
578 000016A6 30E4 <1> xor ah, ah ; 0
579 <1> _sm_10:
580 000016A8 88E0 <1> mov al, ah
581 000016AA 66BACE03 <1> mov dx, 3CEh ; VGAREG_GRDC_ADDRESS
582 000016AE EE <1> out dx, al
583 000016AF AC <1> lodsb
584 000016B0 6642 <1> inc dx ; 3CFh ; VGAREG_GRDC_DATA
585 000016B2 EE <1> out dx, al
586 000016B3 FEC4 <1> inc ah

```



```

587 000016B5 80FC09      <1>      cmp     ah, 9 ; number of grdc regs
588 000016B8 72EE      <1>      jnb     short _sm_10
589                      <1>      ;
590                      <1>      ; Disable CRTC write protection
591 000016BA 66BAD403    <1>      mov     dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
592                      <1>      ;mov     al, 11h
593                      <1>      ;our     dx, al
594                      <1>      ;inc     dx
595                      <1>      ;sub     al, al
596                      <1>      ;out     dx, al
597 000016BE 66B81100    <1>      mov     ax, 11h
598 000016C2 66EF      <1>      out     dx, ax
599 000016C4 89DE      <1>      mov     esi, ebx ; addr of params tbl for selected mode
600 000016C6 83C60A    <1>      add     esi, 10 ; crtc regs
601                      <1>      ; ah = 0
602                      <1>      _sm_11:
603 000016C9 88E0      <1>      mov     al, ah
604                      <1>      ; dx = 3D4h = VGAREG_VGA_CRTC_ADDRESS
605 000016CB EE        <1>      out     dx, al ; index
606 000016CC AC        <1>      lodsb
607 000016CD 6642      <1>      inc     dx ; VGAREG_VGA_CRTC_ADDRESS + 1
608 000016CF EE        <1>      out     dx, al ; value
609 000016D0 80FC18    <1>      cmp     ah, 24 ; number of crtc registers - 1
610 000016D3 7306      <1>      jnb     short _sm_12
611 000016D5 FEC4      <1>      inc     ah
612 000016D7 664A      <1>      dec     dx ; 3D4h
613 000016D9 EBEE      <1>      jmp      short _sm_11
614                      <1>      _sm_12:
615                      <1>      ; Set the misc register
616 000016DB 66BACC03    <1>      mov     dx, 3CCh ; VGAREG_READ_MISC_OUTPUT
617 000016DF 8A4309    <1>      mov     al, [ebx+9] ; misc reg
618 000016E2 EE        <1>      out     dx, al
619                      <1>      ;
620                      <1>      ; Enable video
621 000016E3 66BAC003    <1>      mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
622 000016E7 B020      <1>      mov     al, 20h
623 000016E9 EE        <1>      out     dx, al ; set bit 5 to 1
624 000016EA 66BADA03    <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
625 000016EE EC        <1>      in      al, dx
626                      <1>      ;
627 000016EF 803D[BB450100]00 <1>      cmp     byte [noclearmem], 0
628 000016F6 7740      <1>      ja      short _sm_15
629                      <1>
630                      <1>      ; 29/07/2016
631 000016F8 31C0      <1>      xor     eax, eax
632 000016FA B900400000 <1>      mov     ecx, 4000h ; 16K words (32K)
633 000016FF 803D[D2450100]02 <1>      cmp     byte [VGA_MTYPE], 2 ; CTEXT, MTEXT, CGA
634 00001706 7715      <1>      ja      short _sm_14 ; no ? (0A0000h)
635 00001708 BF00800B00 <1>      mov     edi, 0B8000h
636 0000170D 7409      <1>      je      short _sm_13 ; CGA graphics mode
637                      <1>      ; 08/08/2016
638 0000170F A3[CE450100] <1>      mov     [VGA_INT43H], eax ; 0 ; default font
639 00001714 66B82007    <1>      mov     ax, 0720h ; CGA text mode
640                      <1>      _sm_13:
641 00001718 F366AB      <1>      rep     stosw
642 0000171B EB1B      <1>      jmp      short _sm_15
643                      <1>
644                      <1>      _sm_14:
645 0000171D BF00000A00    <1>      mov     edi, 0A0000h
646                      <1>      ; ecx = 16384 dwords (64K)
647 00001722 66BAC403    <1>      mov     dx, 3C4h ; VGAREG_SEQU_ADDRESS
648 00001726 B002      <1>      mov     al, 2
649 00001728 EE        <1>      out     dx, al
650                      <1>      ;mov     dx, 3C5h ; VGAREG_SEQU_DATA
651 00001729 6642      <1>      inc     dx
652 0000172B EC        <1>      in      al, dx ; mmask
653 0000172C 6650      <1>      push    ax
654 0000172E B00F      <1>      mov     al, 0Fh ; all planes
655 00001730 EE        <1>      out     dx, al
656 00001731 30C0      <1>      xor     al, al ; 0
657 00001733 F3AB      <1>      rep     stosd ; ecx = 163684 (64K)
658 00001735 6658      <1>      pop     ax
659 00001737 EE        <1>      out     dx, al ; mmask
660                      <1>      _sm_15:
661                      <1>      ; ebx = addr of params tbl for selected mode
662                      <1>      ; 10/08/2016
663 00001738 668B03      <1>      mov     ax, [ebx] ; num of columns, 'twidth'
664 0000173B A2[245E0000] <1>      mov     [CRT_COLS], al
665                      <1>      ; 26/07/2016
666                      <1>      ; CRTC_ADDRESS = 3D4h (always)
667                      <1>      ;mov     ah, [ebx+1] ; num of rows, 'theightml'
668 00001740 FEC4      <1>      inc     ah ; 09/07/2016
669 00001742 8825[2A5E0000] <1>      mov     [VGA_ROWS], ah
670                      <1>      ; 10/08/2016
671 00001748 8A4302      <1>      mov     al, [ebx+2]
672 0000174B A2[265E0000] <1>      mov     [CHAR_HEIGHT], al
673                      <1>      ; 29/07/2016
674                      <1>      ; length of regen buffer in bytes
675 00001750 668B4B03      <1>      mov     cx, [ebx+3] ; 'slength_1'
676 00001754 66890D[BC450100] <1>      mov     [CRT_LEN], cx
677                      <1>      ;
678                      <1>      ; 27/07/2016
679 0000175B 30E4      <1>      xor     ah, ah
680 0000175D A0[4E380100] <1>      mov     al, [ACTIVE_PAGE] ; may be > 0 for mode 3
681                      <1>      ;mul     word [CRT_LEN] ; 4096 for mode 3
682 00001762 66F7E1      <1>      mul     cx ; 29/07/2016
683 00001765 66A3[3C380100] <1>      mov     [CRT_START], ax
684                      <1>      ;
685 0000176B B060      <1>      mov     al, 60h
686 0000176D 803D[BB450100]00 <1>      cmp     byte [noclearmem], 0
687 00001774 7602      <1>      jna     short _sm_16
688 00001776 0480      <1>      add     al, 80h

```

[illegible]

```

791      <1>      ;rep    stosw
792      <1>      ;sub    al, al
793 000017F8 EB64      <1>      jmp     short _sm_23
794      <1>      _sm_20:
795      <1>      ; Previous video mode is 3
796      <1>      ; New video mode is 3 while current video mode is not 3
797      <1>      ; (multi screen) video pages will be restored from 0B0000h
798      <1>
799 000017FA 0FB61D[4E380100] <1>      movzx  ebx, byte [ACTIVE_PAGE]
800 00001801 D0E3      <1>      shl     bl, 1 ; * 2
801 00001803 81C3[3E380100] <1>      add     ebx, CURSOR_POSN
802      <1>
803      <1>      ; 29/07/2016
804 00001809 F605[B9450100]7F <1>      test    byte [p_crt_model], 7Fh ; 83h or 3h
805 00001810 7427      <1>      jz      short _sm_21 ; do not restore video pages
806      <1>
807      <1>      ;; restore video pages
808 00001812 BE00800900      <1>      mov     esi, 98000h ; 30/07/2016
809 00001817 BF00800B00      <1>      mov     edi, 0B8000h
810 0000181C 66B90020      <1>      mov     cx, 2000h ; 8K dwords (32K)
811 00001820 F3A5      <1>      rep     movsd
812      <1>
813      <1>      ; restore cursor positions
814 00001822 BE[BE450100] <1>      mov     esi, cursor_pposn
815 00001827 BF[3E380100] <1>      mov     edi, CURSOR_POSN
816      <1>      ;mov    ecx, 4 ; restore all cursor positions (16 bytes)
817 0000182C B104      <1>      mov     cl, 4
818 0000182E F3A5      <1>      rep     movsd
819      <1>
820 00001830 F605[B9450100]80 <1>      test    byte [p_crt_model], 80h
821 00001837 7420      <1>      jz      short _sm_22 ; do not clear current video pages
822      <1>      _sm_21:
823      <1>      ; clear video page
824 00001839 668B0D[BC450100] <1>      mov     cx, [CRT_LEN] ; 4096
825 00001840 66D1E9      <1>      shr     cx, 1 ; 2072
826 00001843 66B82007      <1>      mov     ax, 0720h
827 00001847 BF00800B00      <1>      mov     edi, 0B8000h ; [crt_base]
828 0000184C 66033D[3C380100] <1>      add     di, [CRT_START]
829 00001853 F366AB      <1>      rep     stosw ; FILL THE REGEN BUFFER WITH BLANKS
830      <1>      ;
831 00001856 66890B      <1>      mov     [ebx], cx ; reset cursor position
832      <1>      _sm_22:
833 00001859 A2[B9450100] <1>      mov     [p_crt_model], al ; 0
834      <1>      _sm_23:
835      <1>      ; al = video page number
836      <1>      ; [CRT_LEN] = length of regen buffer in bytes
837 0000185E E81E010000      <1>      call    _set_active_page
838      <1>
839      <1>      ;-----      NORMAL RETURN FROM ALL VIDEO RETURNS
840 00001863 C3      <1>      retn
841      <1>
842      <1>      cursor_shape_fix:
843      <1>      ; 07/07/2016
844      <1>      ; (Cursor start and cursor end line values -6,7-
845      <1>      ; will be fixed depending on character height)
846      <1>      ;
847      <1>      ; derived from 'Plex86/Bochs VGABios' source code
848      <1>      ; vgabios-0.7a (2011)
849      <1>      ; by the LGPL VGABios developers Team (2001-2008)
850      <1>      ; 'vgabios.c', ' biosfn_set_cursor_shape (CH,CL)'
851      <1>      ;
852      <1>      ; INPUT ->
853      <1>      ;     AL = cursor start line (=6)
854      <1>      ;     AH = cursor end line (=7)
855      <1>      ; OUTPUT ->
856      <1>      ;     AL = cursor start line (=14)
857      <1>      ;     AH = cursor end line (=15)
858      <1>      ;
859      <1>      ;; if((modeset_ctl&0x01)&&(cheight>8)&&(CL<8)&&(CH<0x20))
860      <1>
861      <1>      ;test byte [VGA_MODESET_CTL], 1 ; VGA active
862      <1>      ;jz    short csf_3
863 00001864 803D[265E0000]08 <1>      cmp     byte [CHAR_HEIGHT], 8
864 0000186B 7649      <1>      jna     short csf_3
865 0000186D 80FC08      <1>      cmp     ah, 8
866 00001870 7344      <1>      jnb     short csf_3
867 00001872 3C20      <1>      cmp     al, 20h
868 00001874 7340      <1>      jnb     short csf_3
869      <1>      ;
870 00001876 6650      <1>      push    ax
871      <1>      ; {
872      <1>      ; if(CL!=(CH+1))
873 00001878 FEC0      <1>      inc     al
874 0000187A 38C4      <1>      cmp     ah, al ; ah != al + 1
875 0000187C 740F      <1>      je      short csf_1
876      <1>      ; CH = ((CH+1) * cheight / 8) -1;
877 0000187E 8A25[265E0000] <1>      mov     ah, [CHAR_HEIGHT]
878 00001884 F6E4      <1>      mul     ah
879 00001886 C0E803      <1>      shr     al, 3 ; / 8
880 00001889 FEC8      <1>      dec     al ; - 1
881 0000188B EB0E      <1>      jmp     short csf_2
882      <1>      csf_1:
883      <1>      ; }
884      <1>      ; else ; ah = al + 1
885      <1>      ; {
886 0000188D FEC4      <1>      inc     ah ; ah = ah + 1
887      <1>      ; CH = ((CL+1) * cheight / 8) - 2;
888 0000188F A0[265E0000] <1>      mov     al, [CHAR_HEIGHT]
889 00001894 F6E4      <1>      mul     ah
890 00001896 C0E803      <1>      shr     al, 3 ; / 8
891 00001899 2C02      <1>      sub     al, 2 ; - 2
892      <1>      ; al = 14 (if [CHAR_HEIGHT] = 16)

```

```

893                                     <1> csf_2:
894                                     <1>     mov     [esp], al
895                                     <1>     mov     ah, [esp+1]
896                                     <1>     ; CL = ((CL+1) * cheight / 8) - 1;
897                                     <1>     inc     ah
898                                     <1>     mov     al, [CHAR_HEIGHT]
899                                     <1>     mul     ah
900                                     <1>     shr     al, 3 ; / 8
901                                     <1>     dec     al ; - 1
902                                     <1>     mov     [esp+1], al
903                                     <1>     ; ah = 15 (if [CHAR_HEIGHT] = 16)
904                                     <1>     ;
905                                     <1>     pop     ax
906                                     <1> csf_3:
907                                     <1>     retn
908                                     <1>
909                                     <1> SET_CTYPE:
910                                     <1>     ; 12/09/2016
911                                     <1>     ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
912                                     <1>     cmp     byte [CRT_MODE], 7
913                                     <1>     ja      VIDEO_RETURN ; 12/09/2016
914                                     <1>     call    _set_ctype
915                                     <1>     jmp     VIDEO_RETURN
916                                     <1>
917                                     <1> _set_ctype:
918                                     <1>     ; 02/09/2014 (Retro UNIX 386 v1)
919                                     <1>     ;
920                                     <1>     ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
921                                     <1>
922                                     <1>     ; (CH) = BITS 4-0 = START LINE FOR CURSOR
923                                     <1>     ; ** HARDWARE WILL ALWAYS CAUSE BLINK
924                                     <1>     ; ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
925                                     <1>     ; OR NO CURSOR AT ALL
926                                     <1>     ; (CL) = BITS 4-0 = END LINE FOR CURSOR
927                                     <1>
928                                     <1> ;-----
929                                     <1> ; SET_CTYPE
930                                     <1> ; THIS ROUTINE SETS THE CURSOR VALUE
931                                     <1> ; INPUT
932                                     <1> ; (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
933                                     <1> ; OUTPUT
934                                     <1> ; NONE
935                                     <1> ;-----
936                                     <1>
937                                     <1>     ; 07/07/2016
938                                     <1>     ; Fixing cursor start and stop line depending on
939                                     <1>     ; current character height (=16)
940                                     <1>     ; (Note: Default/initial values are 6 and 7.
941                                     <1>     ; If set values are 6 (start) & 7 (stop) and
942                                     <1>     ; [CHAR_HEIGHT] = 16 :
943                                     <1>     ; After fixing, start line will be 14, stop line
944                                     <1>     ; will be 15.)
945                                     <1>     mov     ax, cx
946                                     <1>     xchg    al, ah
947                                     <1>     ; AL = start line, AH = stop line
948                                     <1>     call    cursor_shape_fix
949                                     <1>     ; AL = start line (fixed), AH = stop line (fixed)
950                                     <1>     mov     cx, ax
951                                     <1>     xchg    ch, cl
952                                     <1>     ; CH = start line (fixed), CL = stop line (fixed)
953                                     <1>     ;
954                                     <1>     mov     ah, 10 ; 6845 register for cursor set
955                                     <1>     mov     [CURSOR_MODE], cx ; save in data area
956                                     <1>     ;call    m16 ; output cx register
957                                     <1>     ;retn
958                                     <1>     jmp     m16
959                                     <1>
960                                     <1> SET_CPOS:
961                                     <1>     ; 12/09/2016
962                                     <1>     ; 07/07/2016
963                                     <1>     ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
964                                     <1>     cmp     bh, 7 ; video page > 7 ; 07/07/2016
965                                     <1>     ja      VIDEO_RETURN
966                                     <1>     ;
967                                     <1>     cmp     byte [CRT_MODE], 7
968                                     <1>     ja      short vga_set_cpos ; 12/09/2016
969                                     <1>     call    _set_cpos
970                                     <1>     jmp     VIDEO_RETURN
971                                     <1>
972                                     <1> vga_set_cpos:
973                                     <1>     ; 12/09/2016
974                                     <1>     ; 09/07/2016
975                                     <1>     ; set cursor position
976                                     <1>     ; NOTE: Hardware cursor position will not be set
977                                     <1>     ; in any VGA modes (>7)
978                                     <1>     ; But, cursor position will be saved into
979                                     <1>     ; [CURSOR_POSN].
980                                     <1>     ; TRDOS 386 (TRDOS v2.0) uses only one page
981                                     <1>     ; (page 0) for all graphics modes.
982                                     <1>
983                                     <1>     mov     [CURSOR_POSN], dx ; save cursor pos for pg 0
984                                     <1>     ; 04/08/2016
985                                     <1>     ;mov     bh, [ACTIVE_PAGE] ; = 0
986                                     <1>     ;call    _set_cpos
987                                     <1>     jmp     VIDEO_RETURN
988                                     <1>
989                                     <1> READ_CURSOR:
990                                     <1>     ; 12/09/2016
991                                     <1>     ; 07/07/2016
992                                     <1>     ; 12/05/2016
993                                     <1>     ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
994                                     <1>     ;

```

```

995      <1>      ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
996      <1>
997      <1> ;-----
998      <1> ; READ_CURSOR
999      <1> ;      THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
1000     <1> ;      845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
1001     <1> ; INPUT
1002     <1> ;      BH - PAGE OF CURSOR
1003     <1> ; OUTPUT
1004     <1> ;      DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
1005     <1> ;      CX - CURRENT CURSOR MODE
1006     <1> ;-----
1007     <1>
1008     <1>      ; BH = Video page number (0 to 7)
1009     <1>
1010     <1>      ; 07/07/2016
1011     <1>      cmp     bh, 7 ; video page > 7 (invalid)
1012     <1>      jna     short read_cursor_1
1013     <1>      ; invalid video page (input)
1014     <1>      xor     ecx, ecx ; 0
1015     <1>      xor     edx, edx ; 0
1016     <1>      jmp     short read_cursor_2
1017     <1> read_cursor_1:
1018     <1>      ; 12/09/2016
1019     <1>      cmp     byte [CRT_MODE], 7 ; vga mode
1020     <1>      ja      short vga_get_cpos
1021     <1>      ;
1022     <1>      call    get_cpos
1023     <1>      movzx   ecx, word [CURSOR_MODE]
1024     <1> read_cursor_2:
1025     <1>      pop     ebp
1026     <1>      pop     edi
1027     <1>      pop     esi
1028     <1>      pop     ebx
1029     <1>      pop     eax ; DISCARD SAVED CX AND DX
1030     <1>      pop     eax
1031     <1>      mov     eax, [video_eax] ; 12/05/2016
1032     <1>      ;;15/01/2017
1033     <1>      ;;mov byte [intflg], 0 ; 07/01/2017
1034     <1>      pop     ds
1035     <1>      pop     es
1036     <1>      iretd
1037     <1>
1038     <1> get_cpos:
1039     <1>      ; 12/05/2016
1040     <1>      ; 16/01/2016
1041     <1>      ; BH = Video page number (0 to 7)
1042     <1>      ;
1043     <1>      shl     bh, 1 ; WORD OFFSET
1044     <1>      movzx   esi, bh
1045     <1>      movzx   edx, word [esi+CURSOR_POSN]
1046     <1>      retn
1047     <1>
1048     <1> vga_get_cpos:
1049     <1>      ; 12/09/2016
1050     <1>      ; get cursor position (vga)
1051     <1>      movzx   edx, word [CURSOR_POSN] ; cursor pos for pg 0
1052     <1>      xor     ecx, ecx ; Cursor Mode = 0 (invalid)
1053     <1>      jmp     short read_cursor_2
1054     <1>
1055     <1> ACT_DISP_PAGE:
1056     <1>      ; 07/07/2016
1057     <1>      ; 26/06/2016
1058     <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1059     <1>      ;
1060     <1>      ; VIDEO.ASM - 06/10/85  VIDEO DISPLAY BIOS
1061     <1>      ;
1062     <1> ;-----
1063     <1> ; ACT_DISP_PAGE
1064     <1> ;      THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
1065     <1> ;      THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
1066     <1> ; INPUT
1067     <1> ;      AL HAS THE NEW ACTIVE DISPLAY PAGE
1068     <1> ; OUTPUT
1069     <1> ;      THE 6845 IS RESET TO DISPLAY THAT PAGE
1070     <1> ;-----
1071     <1>      ; 07/07/2016
1072     <1>      cmp     al, 7 ; > 7 = invalid video page number
1073     <1>      ja      VIDEO_RETURN
1074     <1>      cmp     byte [CRT_MODE], 3
1075     <1>      je      short adp_1
1076     <1>      and     al, al
1077     <1>      jnz     VIDEO_RETURN
1078     <1>      ;sub     al, al ; 0 ; force to page 0
1079     <1> adp_1:
1080     <1>      call    set_active_page
1081     <1>      jmp     VIDEO_RETURN
1082     <1>
1083     <1> set_active_page: ; tty_sw
1084     <1>      ; 26/07/2016
1085     <1>      ; 26/06/2016
1086     <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1087     <1>      ; 30/06/2015
1088     <1>      ; 04/03/2014 (act_disp_page --> tty_sw)
1089     <1>      ; 10/12/2013
1090     <1>      ; 04/12/2013
1091     <1>      ;
1092     <1>      mov     [ACTIVE_PAGE], al ; save active page value ; [ptty]
1093     <1> _set_active_page:
1094     <1>      ; 27/06/2015
1095     <1>      movzx   ebx, al
1096     <1>      ;

```



```

1097 00001984 6698 <1> cbw ; 07/09/2014 (ah=0)
1098 00001986 66F725[BC450100] <1> mul word [CRT_LEN] ; get saved length of regen buffer
1099 <1> ; display page times regen length
1100 <1> ; 10/12/2013
1101 0000198D 66A3[3C380100] <1> mov [CRT_START], ax ; save start address for later
1102 00001993 6689C1 <1> mov cx, ax ; start address to cx
1103 <1> _M16:
1104 <1> ;sar cx, 1
1105 00001996 66D1E9 <1> shr cx, 1 ; divide by 2 for 6845 handling
1106 00001999 B40C <1> mov ah, 12 ; 6845 register for start address
1107 0000199B E8D3030000 <1> call m16
1108 <1> ;sal bx, 1
1109 <1> ; 01/09/2014
1110 000019A0 D0E3 <1> shl bl, 1 ; *2 for word offset
1111 000019A2 81C3[3E380100] <1> add ebx, CURSOR_POSN
1112 000019A8 668B13 <1> mov dx, [ebx] ; get cursor for this page
1113 <1> ; 16/01/2016
1114 <1> ;call m18
1115 <1> ;retn
1116 000019AB E9AF030000 <1> jmp m18
1117 <1>
1118 <1> position:
1119 <1> ; 24/06/2016
1120 <1> ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
1121 <1> ; 27/06/2015
1122 <1> ; 02/09/2014
1123 <1> ; 30/08/2014 (Retro UNIX 386 v1)
1124 <1> ; 04/12/2013 (Retro UNIX 8086 v1)
1125 <1> ;
1126 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
1127 <1> ;
1128 <1> ;-----
1129 <1> ; POSITION
1130 <1> ; THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
1131 <1> ; OF A CHARACTER IN THE ALPHA MODE
1132 <1> ; INPUT
1133 <1> ; AX = ROW, COLUMN POSITION
1134 <1> ; OUTPUT
1135 <1> ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
1136 <1> ;-----
1137 <1>
1138 <1> ; DX = ROW, COLUMN POSITION
1139 000019B0 0FB605[245E0000] <1> movzx eax, byte [CRT_COLS] ; 24/06/2016
1140 000019B7 F6E6 <1> mul dh ; row value
1141 000019B9 30F6 <1> xor dh, dh ; 0
1142 000019BB 6601D0 <1> add ax, dx ; add column value to the result
1143 000019BE 66D1E0 <1> shl ax, 1 ; * 2 for attribute bytes
1144 <1> ; EAX = AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
1145 000019C1 C3 <1> retn
1146 <1>
1147 <1> find_position:
1148 <1> ; 24/06/2016
1149 <1> ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
1150 <1> ; 27/06/2015
1151 <1> ; 07/09/2014
1152 <1> ; 02/09/2014
1153 <1> ; 30/08/2014 (Retro UNIX 386 v1)
1154 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
1155 <1>
1156 000019C2 0FB6CF <1> movzx ecx, bh ; video page number
1157 000019C5 89CE <1> mov esi, ecx
1158 000019C7 66D1E6 <1> shl si, 1
1159 000019CA 668B96[3E380100] <1> mov dx, [esi+CURSOR_POSN]
1160 000019D1 740C <1> jz short p21
1161 000019D3 6631F6 <1> xor si, si
1162 <1> p20:
1163 000019D6 660335[BC450100] <1> add si, [CRT_LEN] ; 24/06/2016
1164 <1> ;add si, 80*25*2 ; add length of buffer for one page
1165 000019DD E2F7 <1> loop p20
1166 <1> p21:
1167 000019DF 6621D2 <1> and dx, dx
1168 000019E2 7407 <1> jz short p22
1169 000019E4 E8C7FFFFFF <1> call position ; determine location in regen in page
1170 000019E9 01C6 <1> add esi, eax ; add location to start of regen page
1171 <1> p22:
1172 <1> ;mov dx, [addr_6845] ; get base address of active display
1173 <1> ;mov dx, 03D4h ; I/O address of color card
1174 <1> ;add dx, 6 ; point at status port
1175 000019EB 66BADA03 <1> mov dx, 03DAh ; status port
1176 <1> ; cx = 0
1177 000019EF C3 <1> retn
1178 <1>
1179 <1> SCROLL_UP:
1180 <1> ; 07/07/2016
1181 <1> ; 26/06/2016
1182 <1> ; 12/05/2016
1183 <1> ; 30/01/2016
1184 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1185 <1> ; 07/09/2014
1186 <1> ; 02/09/2014
1187 <1> ; 01/09/2014 (Retro UNIX 386 v1 - beginning)
1188 <1> ; 04/04/2014
1189 <1> ; 04/12/2013
1190 <1> ;
1191 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
1192 <1> ;
1193 <1> ;-----
1194 <1> ; SCROLL UP
1195 <1> ; THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
1196 <1> ; ON THE SCREEN
1197 <1> ; INPUT
1198 <1> ; (AH) = CURRENT CRT MODE

```

```

1199 <1> ; (AL) = NUMBER OF ROWS TO SCROLL
1200 <1> ; (CX) = ROW/COLUMN OF UPPER LEFT CORNER
1201 <1> ; (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
1202 <1> ; (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
1203 <1> ; (DS) = DATA SEGMENT
1204 <1> ; (ES) = REGEN BUFFER SEGMENT
1205 <1> ; OUTPUT
1206 <1> ; NONE -- THE REGEN BUFFER IS MODIFIED
1207 <1> ;-----
1208 <1>
1209 <1> ; 07/07/2016
1210 000019F0 38F5 <1> cmp ch, dh
1211 000019F2 0F875CFBFFFF <1> ja VIDEO_RETURN
1212 000019F8 38D1 <1> cmp cl, dl
1213 000019FA 0F8754FBFFFF <1> ja VIDEO_RETURN
1214 <1> ;
1215 00001A00 E805000000 <1> call _scroll_up
1216 00001A05 E94AFBFFFF <1> jmp VIDEO_RETURN
1217 <1>
1218 <1> _scroll_up: ; from 'write_tty'
1219 <1> ;
1220 <1> ; cl = left upper column
1221 <1> ; ch = left upper row
1222 <1> ; dl = right lower column
1223 <1> ; dh = right lower row
1224 <1> ;
1225 <1> ; al = line count
1226 <1> ; bl = attribute to be used on blanked line
1227 <1> ; bh = video page number (0 to 7)
1228 <1>
1229 00001A0A E896000000 <1> call test_line_count ; 16/01/2016
1230 <1>
1231 00001A0F 8A25[225E0000] <1> mov ah, [CRT_MODE] ; current video mode
1232 <1> ;cmp ah, 4
1233 <1> ;jb short n0
1234 <1> ;cmp byte [CRT_MODE], 4
1235 00001A15 80FC04 <1> cmp ah, 4 ; 07/07/2016
1236 00001A18 0F8320050000 <1> jnb GRAPHICS_UP ; 26/06/2016
1237 <1>
1238 <1> ;cmp ah, 7 ; TEST FOR BW CARD
1239 <1> ;jne GRAPHICS_UP
1240 <1> n0:
1241 <1> ; 07/07/2016
1242 00001A1E 80FF07 <1> cmp bh, 7 ; video page number
1243 00001A21 7606 <1> jna short n1
1244 00001A23 8A3D[4E380100] <1> mov bh, [ACTIVE_PAGE]
1245 <1> n1:
1246 00001A29 88DC <1> mov ah, bl ; attribute
1247 00001A2B 6650 <1> push ax ; *
1248 <1> ;mov esi, [CRT_BASE]
1249 00001A2D BE00800B00 <1> mov esi, 0B8000h
1250 00001A32 3A3D[4E380100] <1> cmp bh, [ACTIVE_PAGE]
1251 00001A38 750B <1> jne short n2
1252 <1> ;
1253 00001A3A 66A1[3C380100] <1> mov ax, [CRT_START]
1254 00001A40 6601C6 <1> add si, ax
1255 00001A43 EB11 <1> jmp short n4
1256 <1> n2:
1257 00001A45 20FF <1> and bh, bh
1258 00001A47 740D <1> jz short n4
1259 00001A49 88F8 <1> mov al, bh
1260 <1> n3:
1261 00001A4B 660335[BC450100] <1> add si, [CRT_LEN]
1262 00001A52 FEC8 <1> dec al
1263 00001A54 75F5 <1> jnz short n3
1264 <1> n4:
1265 00001A56 E85D000000 <1> call scroll_position ; 16/01/2016
1266 00001A5B 7420 <1> jz short n6
1267 <1>
1268 00001A5D 01CE <1> add esi, ecx ; from address for scroll
1269 00001A5F 88F5 <1> mov ch, dh ; #rows in block
1270 00001A61 28C5 <1> sub ch, al ; #rows to be moved
1271 <1> n5:
1272 00001A63 E894000000 <1> call n10 ; 16/01/2016
1273 <1>
1274 00001A68 51 <1> push ecx
1275 00001A69 0FB60D[245E0000] <1> movzx ecx, byte [CRT_COLS]
1276 00001A70 00C9 <1> add cl, cl
1277 00001A72 01CE <1> add esi, ecx ; next line
1278 00001A74 01CF <1> add edi, ecx
1279 00001A76 59 <1> pop ecx
1280 <1>
1281 00001A77 FECF <1> dec ch ; count of lines to move
1282 00001A79 75E8 <1> jnz short n5 ; row loop
1283 <1> ; ch = 0
1284 00001A7B 88C6 <1> mov dh, al ; #rows
1285 <1> n6:
1286 <1> ; attribute in ah
1287 00001A7D B020 <1> mov al, ' ' ; fill with blanks
1288 <1> n7:
1289 00001A7F E885000000 <1> call n11 ; 16/01/2016
1290 <1>
1291 00001A84 8A0D[245E0000] <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[4E380100] <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[235E0000]      <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[245E0000]      <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:      <1>      ; 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>      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:
1394      <1>      ; 07/07/2016
1395      <1>      ; 27/06/2016
1396      <1>      ; 26/06/2016
1397      <1>      ; 12/05/2016
1398      <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1399      <1>      ;
1400      <1>      ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
1401      <1>
1402      <1> ;-----

```

```

1403 <1> ; SCROLL DOWN
1404 <1> ; THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
1405 <1> ; BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
1406 <1> ; WITH A DEFINED CHARACTER
1407 <1> ; INPUT
1408 <1> ; (AH) = CURRENT CRT MODE
1409 <1> ; (AL) = NUMBER OF LINES TO SCROLL
1410 <1> ; (CX) = UPPER LEFT CORNER OF RECIION
1411 <1> ; (DX) = LOWER RIGHT CORNER OF REGION
1412 <1> ; (BH) = FILL CHARACTER
1413 <1> ; (DS) = DATA SEGMENT
1414 <1> ; (ES) = REGEN SEGMENT
1415 <1> ; OUTPUT
1416 <1> ; NONE -- SCREEN IS SCROLLED
1417 <1> ;-----
1418 <1>
1419 <1> ; 07/07/2016
1420 00001B14 38F5 <1> cmp ch, dh
1421 00001B16 0F8738FAFFFF <1> ja VIDEO_RETURN
1422 00001B1C 38D1 <1> cmp cl, dl
1423 00001B1E 0F8730FAFFFF <1> ja VIDEO_RETURN
1424 <1> ;
1425 00001B24 E805000000 <1> call _scroll_down
1426 00001B29 E926FAFFFF <1> jmp VIDEO_RETURN
1427 <1>
1428 <1> _scroll_down: ; 27/06/2016
1429 <1>
1430 <1> ; cl = left upper column
1431 <1> ; ch = left upper row
1432 <1> ; dl = right lower column
1433 <1> ; dh = right lower row
1434 <1> ;
1435 <1> ; al = line count
1436 <1> ; bl = attribute to be used on blanked line
1437 <1> ; bh = video page number (0 to 7)
1438 <1>
1439 <1> ; !!!!
1440 00001B2E FD <1> std ; DIRECTION FOR SCROLL DOWN
1441 <1> ; !!!!
1442 00001B2F E871FFFFFF <1> call test_line_count ; 16/01/2016
1443 <1>
1444 00001B34 8A25[225E0000] <1> mov ah, [CRT_MODE] ; current video mode
1445 <1> ;cmp ah, 4
1446 <1> ;jnb short _n0
1447 <1> ;cmp byte [CRT_MODE], 4
1448 00001B3A 80FC04 <1> cmp ah, 4 ; 07/07/2016
1449 00001B3D 0F83DF070000 <1> jnb GRAPHICS_DOWN ; 26/06/2016
1450 <1>
1451 <1> ;cmp ah, 7 ; TEST FOR BW CARD
1452 <1> ;jne GRAPHICS_DOWN
1453 <1> _n0:
1454 <1> ; 07/07/2016
1455 00001B43 80FF07 <1> cmp bh, 7 ; video page number
1456 00001B46 7606 <1> jna short n12
1457 00001B48 8A3D[4E380100] <1> mov bh, [ACTIVE_PAGE]
1458 <1> ;
1459 <1> n12: ; CONTINUE_DOWN
1460 00001B4E 88DC <1> mov ah, bl
1461 00001B50 6650 <1> push ax ; * ; save attribute in ah
1462 00001B52 6689D0 <1> mov ax, dx ; LOWER RIGHT CORNER
1463 00001B55 E85EFFFFFF <1> call scroll_position ; GET REGEN LOCATION
1464 00001B5A 741F <1> jz short n14
1465 00001B5C 29CE <1> sub esi, ecx ; SI IS FROM ADDRESS
1466 00001B5E 88F5 <1> mov ch, dh ; #rows in block
1467 00001B60 28C5 <1> sub ch, al ; #rows to be moved
1468 <1> n13:
1469 00001B62 E895FFFFFF <1> call n10 ; MOVE ONE ROW
1470 <1>
1471 00001B67 51 <1> push ecx
1472 00001B68 8A0D[245E0000] <1> mov cl, [CRT_COLS]
1473 00001B6E 00C9 <1> add cl, cl
1474 00001B70 29CE <1> sub esi, ecx ; next line
1475 00001B72 29CF <1> sub edi, ecx
1476 00001B74 59 <1> pop ecx
1477 <1>
1478 00001B75 FECD <1> dec ch ; count of lines to move
1479 00001B77 75E9 <1> jnz short n13 ; row loop
1480 <1> ; ch = 0
1481 00001B79 88C6 <1> mov dh, al ; #rows
1482 <1> n14:
1483 <1> ; attribute in ah
1484 00001B7B B020 <1> mov al, ' ' ; fill with blanks
1485 <1> n15:
1486 00001B7D E887FFFFFF <1> call n11 ; 16/01/2016
1487 <1>
1488 00001B82 8A0D[245E0000] <1> mov cl, [CRT_COLS]
1489 00001B88 00C9 <1> add cl, cl
1490 00001B8A 29CF <1> sub edi, ecx
1491 <1>
1492 00001B8C FECE <1> dec dh
1493 00001B8E 75ED <1> jnz short n15
1494 <1> ;
1495 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[225E0000]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[225E0000]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[225E0000] <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[4E380100] <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> ; THIS 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[225E0000]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[2B5E0000] <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[225E0000]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[2B5E0000] <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[225E0000]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[225E0000] <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
1710 00001C68 FB
1711 00001C69 3A3D[4E380100]
1712 00001C6F 7510
1713 00001C71 FA
1714 00001C72 EC
1715 00001C73 A808
1716 00001C75 7509
1717 00001C77 A801
1718 00001C79 75ED
1719
1720 00001C7B EC
1721 00001C7C A809
1722 00001C7E 74FB
1723
1724 00001C80 FB
1725
1726 00001C81 668B0424
1727 00001C85 668906
1728
1729 00001C88 6649
1730 00001C8A 7404
1731
1732 00001C8C 46
1733 00001C8D 46
1734 00001C8E EBD8
1735
1736 00001C90 6658
1737 00001C92 665A
1738 00001C94 C3
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776 00001C95 803D[225E0000]07
1777 00001C9C 760A
1778
1779 00001C9E E8290D0000
1780 00001CA3 E9ACF8FFFF
1781
1782
1783
1784
1785
1786 00001CA8 E818000000
1787 00001CAD E9A2F8FFFF
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810

<1> p41:
<1>      sti          ; enable interrupts first
<1>      cmp          bh, [ACTIVE_PAGE]
<1>      jne          short p44
<1>      cli          ; block interrupts for single loop
<1>      in            al, dx ; get status from the adapter
<1>      test         al, RVRT ; check for vertical retrace first
<1>      jnz          short p43 ; Do fast write now if vertical retrace
<1>      test         al, RHRZ ; is horizontal retrace low
<1>      jnz          short p41 ; wait until it is
<1> p42:
<1>      in            al, dx ; get status again
<1>      test         al, RVRT+RHRZ ; is horizontal or vertical retrace high
<1>      jz           short p42 ; wait until either retrace active
<1> p43:
<1>      sti
<1> p44:
<1>      mov          ax, [esp] ; restore the character (al) & attribute (ah)
<1>      mov          [esi], ax
<1>
<1>      dec          cx
<1>      jz           short p45
<1>
<1>      inc          esi
<1>      inc          esi
<1>      jmp          short p41
<1> p45:
<1>      pop          ax
<1>      pop          dx
<1>      retn
<1>
<1> ; 09/07/2016
<1> ; 26/06/2016
<1> ; 24/06/2016
<1> ; 12/05/2016
<1> ; 18/01/2016
<1> ; 16/01/2016 - TRDOS 386 (TRDOS v2.0)
<1> ; 30/06/2015
<1> ; 27/06/2015
<1> ; 11/03/2015
<1> ; 02/09/2014
<1> ; 30/08/2014
<1> ; VIDEO FUNCTIONS
<1> ; (write_tty - Retro UNIX 8086 v1 - U9.ASM, 01/02/2014)
<1>
<1> WRITE_TTY:
<1>      ; 09/07/2016
<1>      ; 01/07/2016
<1>      ; 26/06/2016
<1>      ; 24/06/2016
<1>      ; 13/05/2016
<1>      ; 12/05/2016
<1>      ; 30/01/2016
<1>      ; 18/01/2016
<1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
<1>      ; 13/08/2015
<1>      ; 02/09/2014
<1>      ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
<1>      ; 01/02/2014 (Retro UNIX 8086 v1 - last update)
<1>      ; 03/12/2013 (Retro UNIX 8086 v1 - beginning)
<1>      ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
<1>      ;
<1>      ; INPUT -> AL = Character to be written
<1>      ;          BL = Color (Forecolor, Backcolor)
<1>      ;          BH = Video Page (0 to 7)
<1>
<1>      ; 09/07/2016
<1>      cmp          byte [CRT_MODE], 7
<1>      jna          short write_tty_cga
<1>
<1>      call         vga_write_teletype
<1>      jmp          VIDEO_RETURN
<1>
<1> write_tty_cga:
<1>      ; 13/05/2016
<1>      ;call _write_tty
<1>      ; 01/07/2016
<1>      call         _write_tty_m3
<1>      jmp          VIDEO_RETURN
<1>
<1> RVRT equ          00001000b ; VIDEO VERTICAL RETRACE BIT
<1> RHRZ equ          00000001b ; VIDEO HORIZONTAL RETRACE BIT
<1>
<1> ; Derived from "WRITE_TTY" procedure of IBM "pc-at" rombios source code
<1> ; (06/10/1985), 'video.asm', INT 10H, VIDEO_IO
<1> ;
<1> ; 06/10/85 VIDEO DISPLAY BIOS
<1> ;
<1> ;--- WRITE_TTY -----
<1> ;
<1> ;
<1> ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
<1> ; VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
<1> ; CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
<1> ; IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
<1> ; IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW
<1> ; ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW,
<1> ; FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE.
<1> ; WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE
<1> ; NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS
<1> ; LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE,
<1> ; THE 0 COLOR IS USED.
<1> ;
<1> ; ENTRY --

```

```

1811 <1> ; (AH) = CURRENT CRT MODE :
1812 <1> ; (AL) = CHARACTER TO BE WRITTEN :
1813 <1> ; NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE :
1814 <1> ; HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS :
1815 <1> ; (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE :
1816 <1> ; EXIT -- :
1817 <1> ; ALL REGISTERS SAVED :
1818 <1> ;-----
1819 <1>
1820 <1> ; 08/07/2016
1821 <1> ; 26/06/2016
1822 <1> ; 24/06/2016
1823 <1> _write_tty: ; 13/05/2016
1824 00001CB2 FA <1> cli
1825 <1> ;
1826 <1> ; 01/09/2014
1827 00001CB3 803D[225E0000]03 <1> cmp byte [CRT_MODE], 3
1828 00001CBA 7409 <1> je short _write_tty_m3
1829 <1> ;
1830 <1> set_mode_3:
1831 00001CBC 53 <1> push ebx
1832 00001CBD 50 <1> push eax
1833 00001CBE E8A2F8FFFF <1> call _set_mode
1834 00001CC3 58 <1> pop eax
1835 00001CC4 5B <1> pop ebx
1836 <1> ;
1837 <1> _write_tty_m3: ; 24/06/2016 (m3 -> _write_tty_m3)
1838 00001CC5 0FB6F7 <1> movzx esi, bh ; 12/05/2016
1839 00001CC8 66D1E6 <1> shl si, 1
1840 00001CCB 81C6[3E380100] <1> add esi, CURSOR_POSN
1841 00001CD1 668B16 <1> mov dx, [esi]
1842 <1> ;
1843 <1> ; dx now has the current cursor position
1844 <1> ;
1845 00001CD4 3C0D <1> cmp al, 0Dh ; CR ; is it carriage return or control character
1846 00001CD6 7636 <1> jbe short u8
1847 <1> ;
1848 <1> ; write the char to the screen
1849 <1> u0:
1850 <1> ; al = character
1851 <1> ; bl = attribute/color
1852 <1> ; bh = video page number (0 to 7)
1853 <1> ;
1854 00001CD8 66B90100 <1> mov cx, 1 ; 24/06/2016
1855 <1> ; cx = count of characters to write
1856 <1> ;
1857 00001CDC E858FFFFFF <1> call _write_c_current ; 16/01/2015
1858 <1> ;
1859 <1> ; position the cursor for next char
1860 00001CE1 FEC2 <1> inc dl ; next column
1861 00001CE3 3A15[245E0000] <1> cmp dl, [CRT_COLS] ; test for column overflow
1862 00001CE9 755D <1> jne _set_cpos
1863 00001CEB B200 <1> mov dl, 0 ; column = 0
1864 <1> u10: ; (line feed found)
1865 00001CED 80FE18 <1> cmp dh, 25-1 ; check for last row
1866 00001CF0 7218 <1> jb short u6
1867 <1> ;
1868 <1> ; scroll required
1869 <1> u1:
1870 <1> ; SET CURSOR POSITION (04/12/2013)
1871 00001CF2 E851000000 <1> call _set_cpos
1872 <1> ;
1873 <1> ; determine value to fill with during scroll
1874 <1> u2:
1875 <1> ; bh = video page number
1876 <1> ;
1877 00001CF7 E8B3FFFFFF <1> call _read_ac_current ; 18/01/2016
1878 <1> ;
1879 <1> ; al = character, ah = attribute
1880 <1> ; bh = video page number
1881 <1> u3:
1882 <1> ;mov ax, 0601h ; scroll one line
1883 <1> ;sub cx, cx ; upper left corner
1884 <1> ;mov dh, 25-1 ; lower right row
1885 <1> ;mov dl, [CRT_COLS]
1886 <1> ;mov dl, 80 ; lower right column
1887 <1> ;dec dl
1888 <1> ;mov dl, 79
1889 <1>
1890 <1> ;call scroll_up ; 04/12/2013
1891 <1> ; 11/03/2015
1892 <1> ; 02/09/2014
1893 <1> ;mov cx, [crt_ulc] ; Upper left corner (0000h)
1894 <1> ;mov dx, [crt_lrc] ; Lower right corner (184Fh)
1895 <1> ; 11/03/2015
1896 00001CFC 6629C9 <1> sub cx, cx
1897 00001CFF 66BA4F18 <1> mov dx, 184Fh ; dl = 79 (column), dh = 24 (row)
1898 <1> ;
1899 00001D03 B001 <1> mov al, 1 ; scroll 1 line up
1900 <1> ; ah = attribute
1901 <1> ;mov bl, al ; 12/05/2016
1902 00001D05 E900FDFFFF <1> jmp _scroll_up ; 16/01/2016
1903 <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:
1911 00001D0A FEC6 <1> inc dh ; set-cursor-inc
1912 <1> ; next row
; set cursor

```

```

1913
1914
1915
1916
1917
1918 00001D0C EB3A
1919
1920
1921
1922 00001D0E 7436
1923 00001D10 3C0A
1924 00001D12 74D9
1925 00001D14 3C07
1926 00001D16 747A
1927 00001D18 3C08
1928
1929 00001D1A 7422
1930
1931 00001D1C 3C09
1932 00001D1E 75B8
1933 00001D20 88D0
1934 00001D22 6698
1935 00001D24 B108
1936 00001D26 F6F1
1937 00001D28 28E1
1938
1939
1940
1941 00001D2A B020
1942
1943 00001D2C 6651
1944 00001D2E 6650
1945
1946 00001D30 E890FFFFFF
1947 00001D35 6658
1948 00001D37 6659
1949 00001D39 FEC9
1950 00001D3B 75EF
1951 00001D3D C3
1952
1953
1954
1955 00001D3E 08D2
1956
1957 00001D40 7406
1958 00001D42 664A
1959
1960 00001D44 EB02
1961
1962
1963
1964 00001D46 B200
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995 00001D48 BE[3E380100]
1996 00001D4D 0FB6C7
1997
1998
1999 00001D50 D0E0
2000 00001D52 01C6
2001
2002 00001D54 668916
2003 00001D57 383D[4E380100]
2004 00001D5D 7532
2005
2006
2007
2008
2009
2010
2011 00001D5F E84CFCFFFF
2012 00001D64 668B0D[3C380100]
2013 00001D6B 6601C1
2014

<1> ;u7:
<1> ;mov ah, 02h
<1> ;jmp short u4 ; establish the new cursor
<1> ;call _set_cpos
<1> ;jmp short u5
<1> jmp _set_cpos
<1>
<1> ; check for control characters
<1> u8:
<1> je short u9
<1> cmp al, 0Ah ; is it a line feed (0Ah)
<1> je short u10
<1> cmp al, 07h ; is it a bell
<1> je short u11
<1> cmp al, 08h ; is it a backspace
<1> ;jne short u0
<1> je short bs ; 12/12/2013
<1> ; 12/12/2013 (tab stop)
<1> cmp al, 09h ; is it a tab stop
<1> jne short u0
<1> mov al, dl
<1> cbw
<1> mov cl, 8
<1> div cl
<1> sub cl, ah
<1> ts:
<1> ; 02/09/2014
<1> ; 01/09/2014
<1> mov al, 20h
<1> tsloop:
<1> push cx
<1> push ax
<1> ;mov bh, [ACTIVE_PAGE]
<1> call _write_tty_m3 ; 24/06/2016 (m3 -> _write_tty_m3)
<1> pop ax ; ah = attribute/color
<1> pop cx
<1> dec cl
<1> jnz short tsloop
<1> retn
<1> bs:
<1> ; back space found
<1>
<1> or dl, dl ; is it already at start of line
<1> ;je short u7 ; set_cursor
<1> jz short _set_cpos
<1> dec dx ; no -- just move it back
<1> ;jmp short u7
<1> jmp short _set_cpos
<1>
<1> ; carriage return found
<1> u9:
<1> mov dl, 0 ; move to first column
<1> ;jmp short u7
<1> ;jmp short _set_cpos ; 30/01/2016
<1>
<1> ; line feed found
<1> ;u10:
<1> ; cmp dh, 25-1 ; bottom of screen
<1> ; jne short u6 ; no, just set the cursor
<1> ; jmp ul ; yes, scroll the screen
<1>
<1> _set_cpos:
<1> ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
<1> ; 27/06/2015
<1> ; 01/09/2014
<1> ; 30/08/2014 (Retro UNIX 386 v1)
<1> ;
<1> ; 04/12/2013 - 12/12/2013 (Retro UNIX 8086 v1)
<1> ;
<1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
<1> ;
<1> ;-----
<1> ; SET_CPOS
<1> ; THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
<1> ; NEW X-Y VALUES PASSED
<1> ; INPUT
<1> ; DX - ROW,COLUMN OF NEW CURSOR
<1> ; BH - DISPLAY PAGE OF CURSOR
<1> ; OUTPUT
<1> ; CURSOR ID SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
<1> ;-----
<1> ;
<1> mov esi, CURSOR_POSN
<1> movzx eax, bh ; BH = video page number
<1> ; or al, al
<1> ; jz short _set_cpos_0
<1> shl al, 1 ; word offset
<1> add esi, eax
<1> ;_set_cpos_0:
<1> mov [esi], dx ; save the pointer
<1> cmp [ACTIVE_PAGE], bh
<1> jne short m17
<1> ;call m18 ; CURSOR SET
<1> ;m17: ; SET_CPOS_RETURN
<1> ; 01/09/2014
<1> ; retn
<1> ; DX = row/column
<1> m18:
<1> call position ; determine location in regen buffer
<1> mov cx, [CRT_START]
<1> add cx, ax ; add char position in regen buffer
<1> ; to the start address (offset) for this page

```

```

2015 00001D6E 66D1E9      <1>      shr    cx, 1 ; divide by 2 for char only count
2016 00001D71 B40E      <1>      mov    ah, 14 ; register number for cursor
2017                      <1>      ;call m16 ; output value to the 6845
2018                      <1>      ;retn
2019                      <1>
2020                      <1>      ;----- THIS ROUTINE OUTPUTS THE CX REGISTER
2021                      <1>      ;      TO THE 6845 REGISTERS NAMED IN (AH)
2022                      <1> m16:
2023 00001D73 FA        <1>      cli
2024                      <1>      ;mov dx, [addr_6845] ; address register
2025 00001D74 66BAD403   <1>      mov    dx, 03D4h ; I/O address of color card
2026 00001D78 88E0      <1>      mov    al, ah ; get value
2027 00001D7A EE        <1>      out    dx, al ; register set
2028 00001D7B 6642      <1>      inc    dx ; data register
2029 00001D7D EB00      <1>      jmp    $+2 ; i/o delay
2030 00001D7F 88E8      <1>      mov    al, ch ; data
2031 00001D81 EE        <1>      out    dx, al
2032 00001D82 664A      <1>      dec    dx
2033 00001D84 88E0      <1>      mov    al, ah
2034 00001D86 FEC0      <1>      inc    al ; point to other data register
2035 00001D88 EE        <1>      out    dx, al ; set for second register
2036 00001D89 6642      <1>      inc    dx
2037 00001D8B EB00      <1>      jmp    $+2 ; i/o delay
2038 00001D8D 88C8      <1>      mov    al, cl ; second data value
2039 00001D8F EE        <1>      out    dx, al
2040 00001D90 FB        <1>      sti
2041                      <1> m17:
2042 00001D91 C3        <1>      retn
2043                      <1>
2044                      <1> beeper:
2045                      <1>      ; 04/08/2016
2046                      <1>      ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
2047                      <1>      ; 30/08/2014 (Retro UNIX 386 v1)
2048                      <1>      ; 18/01/2014
2049                      <1>      ; 03/12/2013
2050                      <1>      ; bell found
2051                      <1> u11:
2052 00001D92 FB        <1>      sti
2053 00001D93 3A3D[4E380100] <1>      cmp    bh, [ACTIVE_PAGE]
2054 00001D99 7551      <1>      jne    short u12 ; Do not sound the beep
2055                      <1>      ; if it is not written on the active page
2056                      <1> beeper_gfx: ; 04/08/2016
2057 00001D9B 66B93305   <1>      mov    cx, 1331 ; divisor for 896 hz tone
2058 00001D9F B31F      <1>      mov    bl, 31 ; set count for 31/64 second for beep
2059                      <1>      ;call beep ; sound the pod bell
2060                      <1>      ;jmp short u5 ; tty_return
2061                      <1>      ;retn
2062                      <1>
2063                      <1> TIMER equ 040h ; 8254 TIMER - BASE ADDRESS
2064                      <1> PORT_B equ 061h ; PORT B READ/WRITE DIAGNOSTIC REGISTER
2065                      <1> GATE2 equ 00000001b ; TIMER 2 INPUT CATE CLOCK BIT
2066                      <1> SPK2 equ 00000010b ; SPEAKER OUTPUT DATA ENABLE BIT
2067                      <1>
2068                      <1> beep:
2069                      <1>      ; 07/02/2015
2070                      <1>      ; 30/08/2014 (Retro UNIX 386 v1)
2071                      <1>      ; 18/01/2014
2072                      <1>      ; 03/12/2013
2073                      <1>      ;
2074                      <1>      ; TEST4.ASM - 06/10/85 POST AND BIOS UTILITY ROUTINES
2075                      <1>      ;
2076                      <1>      ; ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE
2077                      <1>      ;
2078                      <1>      ; ENTRY:
2079                      <1>      ; (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )
2080                      <1>      ; (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)
2081                      <1>      ; EXIT:
2082                      <1>      ; (AX),(BL),(CX) MODIFIED.
2083                      <1>
2084 00001DA1 9C        <1>      pushf ; 18/01/2014 ; save interrupt status
2085 00001DA2 FA        <1>      cli ; block interrupts during update
2086 00001DA3 B0B6      <1>      mov    al, 10110110b ; select timer 2, lsb, msb binary
2087 00001DA5 E643      <1>      out    TIMER+3, al ; write timer mode register
2088 00001DA7 EB00      <1>      jmp    $+2 ; I/O delay
2089 00001DA9 88C8      <1>      mov    al, cl ; divisor for hz (low)
2090 00001DAB E642      <1>      out    TIMER+2,AL ; write timer 2 count - lsb
2091 00001DAD EB00      <1>      jmp    $+2 ; I/O delay
2092 00001DAF 88E8      <1>      mov    al, ch ; divisor for hz (high)
2093 00001DB1 E642      <1>      out    TIMER+2, al ; write timer 2 count - msb
2094 00001DB3 E461      <1>      in     al, PORT_B ; get current setting of port
2095 00001DB5 88C4      <1>      mov    ah, al ; save that setting
2096 00001DB7 0C03      <1>      or     al, GATE2+SPK2 ; gate timer 2 and turn speaker on
2097 00001DB9 E661      <1>      out    PORT_B, al ; and restore interrupt status
2098                      <1>      ;popf ; 18/01/2014
2099 00001DBB FB        <1>      sti
2100                      <1> g7:
2101 00001DBC B90B040000 <1>      mov    ecx, 1035 ; 1/64 second per count (bl)
2102 00001DC1 E827000000 <1>      call   waitf ; delay count for 1/64 of a second
2103 00001DC6 FECB      <1>      dec    bl ; go to beep delay 1/64 count
2104 00001DC8 75F2      <1>      jnz    short g7 ; (bl) length count expired?
2105                      <1>      ; no - continue beeping speaker
2106                      <1>      ;
2107 00001DCA FA        <1>      ;pushf ; save interrupt status
2108 00001DCB E461      <1>      cli ; 18/01/2014 ; block interrupts during update
2109                      <1>      in     al, PORT_B ; get current port value
2110                      <1>      ;or al, not (GATE2+SPK2) ; isolate current speaker bits in case
2111                      <1>      ;or al, ~(GATE2+SPK2)
2112                      <1>      and    ah, al ; someone turned them off during beep
2113                      <1>      mov    al, ah ; recover value of port
2114                      <1>      ;or al, not (GATE2+SPK2) ; force speaker data off
2115                      <1>      or     al, ~(GATE2+SPK2) ; isolate current speaker bits in case
2116                      <1>      out    PORT_B, al ; and stop speaker timer
2117                      <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> ul2:
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 - AWARD BIOS 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[B8450100] <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 E8FCC90000 <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[225E0000]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[3E380100] <1> PUSH word [esi+Cursors_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[3E380100] <1> MOV DX, [esi+Cursors_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[B8450100]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[2B5E0000] <1> mov [esi+chr_attrib], bl ; color/attribute
2313 <1>
2314 00001E95 FEC2 <1> INC DL ; INCREMENT COLUMN COUNTER
2315 00001E97 3A15[245E0000] <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[B8450100]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[3E380100] <1> mov [CURSOR_POSN], dx ; save cursor pos for pg 0
2397 <1> vga_wstr_1:
2398 00001EE1 66FF35[3E380100] <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[B8450100]02 <1> test byte [w_str_cmd], 2
2413 00001EEF 751D <1> jnz short vga_wstr_3
2414 00001EF1 881D[4F380100] <1> mov [ccolor], bl
2415 <1> vga_wstr_2:
2416 <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[4F380100] <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[B8450100]01 <1>      test    byte [w_str_cmd], 1
2445 00001F2C 0F8522F6FFFF <1>      jnz     VIDEO_RETURN
2446 00001F32 668915[3E380100] <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 righth 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[225E0000]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              ; EVERYTHING 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 righth 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[465E0000] <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>          jnb     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[2A5E0000] <1>          cmp     dh, [VGA_ROWS]
2589 00001FCA 7208      <1>          jnb     short vga_g_up_2
2590 00001FCC 8A35[2A5E0000] <1>          mov     dh, [VGA_ROWS]
2591 00001FD2 FECE      <1>          dec     dh
2592                    <1> vga_g_up_2:
2593 00001FD4 3A15[245E0000] <1>          cmp     dl, [CRT_COLS] ; = [VGA_COLS]
2594 00001FDA 7208      <1>          jnb     short vga_g_up_3
2595 00001FDC 8A15[245E0000] <1>          mov     dl, [CRT_COLS]
2596 00001FE2 FECA      <1>          dec     dl
2597                    <1> vga_g_up_3:
2598 00001FE4 3A05[2A5E0000] <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[2A5E0000] <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[245E0000] <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[2A5E0000] <1> mov al, [VGA_ROWS]
2628 00002029 8A25[245E0000] <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[265E0000] <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 righth 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,height);
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,height,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 righth 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,height);
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,height,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,height)
2733      <1>      ; cl = xstart, ah = ysrc (i+nblines), ch = ydest (i),
2734      <1>      ; bh = cols, [CRT_COLS] = nbcols, [CHAR_HEIGHT] = height
2735      <1>
2736      <1>      ; src=ysrc*height*nbcols+xstart;
2737      <1>      ; dest=ydest*height*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[265E0000] <1>      movzx  eax, byte [CHAR_HEIGHT]
2753      000020B5 8A25[245E0000] <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 ; height * nbcols
2759      000020BE F7E2      <1>      mul     edx ; * ysrc
2760      <1>      ; eax = ysrc * height * 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 ; height * nbcols
2767      000020C9 F7E2      <1>      mul     edx
2768      <1>      ; eax = ydest * height * 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<height;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[265E0000] <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[245E0000] <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,height,attr)
2826      <1>      ; cl = xstart, edi = ch = ystart, bh = cols,
2827      <1>      ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = height, attr = 0
2828      <1>
2829      <1>      ; dest=ystart*height*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[265E0000] <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[245E0000] <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[265E0000] <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[245E0000] <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[265E0000] <1>      movzx   eax, byte [CHAR_HEIGHT]
2919 0000216E 8A25[245E0000] <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[265E0000] <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[245E0000] <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[265E0000] <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[245E0000] <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[265E0000] <1>      mov     dh, [CHAR_HEIGHT]
3034                  <1>      ; 07/08/2016

```

```

3035          ;mov  dh, 8 ; 07/08/2016
3036 000021F7 28D2          sub  dl, dl ; i
3037          <1> vgamem_fill_l8_0:
3038 000021F9 57            push  edi
3039 000021FA 0FB605[245E0000] movzx  eax, byte [CRT_COLS]
3040 00002201 F6E2          mul   dl
3041          <1>          ; eax = i * nbcols
3042 00002203 66C1E003      shl   ax, 3 ; * 8 ; 06/08/2016
3043 00002207 01C7          add   edi, eax ; dest+i*nbcols
3044 00002209 88D8          mov   al, bl ; attr ; 04/08/2016
3045 0000220B 0FB6CF      movzx  ecx, bh ; cols
3046 0000220E 66C1E103    shl   cx, 3 ; * 8 ; 06/08/2016
3047 00002212 F3AA          rep   stosb
3048 00002214 5F            pop   edi
3049 00002215 FEC2          inc   dl ; 06/08/2016
3050 00002217 FECE          dec   dh
3051 00002219 75DE          jnz   short vgamem_fill_l8_0
3052          <1> vgamem_fill_l8_1:
3053 0000221B 59            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            pop   eax
3061 0000221D 5A            pop   edx
3062          <1>
3063 0000221E C3            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 righth 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            cld      ; !!! Clear direction flag !!!
3089          <1>
3090 00002220 88C7          mov   bh, al ; 31/07/2016
3091          <1>
3092 00002222 BE[3E5E0000]   mov   esi, vga_modes
3093 00002227 89F7          mov   edi, esi
3094 00002229 83C710        add   edi, vga_mode_count
3095          <1> vga_g_down_0:
3096 0000222C AC            lodsb
3097 0000222D 38E0          cmp   al, ah ; [CRT_MODE]
3098 0000222F 7405          je    short vga_g_down_1
3099 00002231 39FE          cmp   esi, edi
3100 00002233 72F7          jb    short vga_g_down_0
3101          <1>          ; xor  bh, bh ; 31/07/2016
3102 00002235 C3            retn   ; nothing to do
3103          <1> vga_g_down_1:
3104 00002236 88F8          mov   al, bh ; 31/07/2016
3105 00002238 83C64F        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[2A5E0000]   cmp   dh, [VGA_ROWS]
3114 00002241 7208          jb    short vga_g_down_2
3115 00002243 8A35[2A5E0000]   mov   dh, [VGA_ROWS]
3116 00002249 FECE          dec   dh
3117          <1> vga_g_down_2:
3118 0000224B 3A15[245E0000]   cmp   dl, [CRT_COLS] ; = [VGA_COLS]
3119 00002251 7208          jb    short vga_g_down_3
3120 00002253 8A15[245E0000]   mov   dl, [CRT_COLS]
3121 00002259 FECA          dec   dl
3122          <1> vga_g_down_3:
3123 0000225B 3A05[2A5E0000]   cmp   al, [VGA_ROWS]
3124 00002261 7602          jna    short vga_g_down_4
3125 00002263 28C0          sub   al, al ; 0
3126          <1> vga_g_down_4:
3127 00002265 88F7          mov   bh, dh ; clr
3128 00002267 28CF          sub   bh, cl ; cul
3129 00002269 FEC7          inc   bh ; cols = clr-cul+1
3130          <1>
3131 0000226B 20C0          and   al, al ; nblines = 0
3132 0000226D 755B          jnz    short vga_g_down_6
3133 0000226F 20ED          and   ch, ch ; rul = 0
3134 00002271 7557          jnz    short vga_g_down_6
3135 00002273 20C9          and   cl, cl ; cul = 0
3136 00002275 7553          jnz    short vga_g_down_6

```



```

3137                                     <1>
3138 00002277 6650                       <1>      push    ax
3139 00002279 A0[2A5E0000]               <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[245E0000]               <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[2A5E0000]               <1>      mov     al, [VGA_ROWS]
3153 000022A0 8A25[245E0000]             <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[265E0000]           <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 righth 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>      jnb     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,height);
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,height,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 righth column ; clr
3228                                     <1>      ; dh = lower right row ; rlr
3229                                     <1> vga_g_down_pl0:
3230                                     <1>      ;{for(i=rlr;i>=rul;i--)
3231                                     <1>      ; if((i<rul+nblines)|| (nblines==0))
3232 000022FB 08C0                       <1>      or      al, al
3233 000022FD 741C                       <1>      jz      short vga_g_down_pl2
3234 000022FF 88C4                       <1>      mov     ah, al
3235 00002301 00EC                       <1>      add     ah, ch
3236                                     <1>      ;jc     short vga_g_down_pl2
3237 00002303 86EE                       <1>      xchg    ch, dh
3238 00002305 38E5                       <1>      cmp     ch, ah

```

```

3239 00002307 7212      <1>      jb      short vga_g_down_pl2
3240 00002309 88EC      <1>      mov     ah, ch
3241 0000230B 28C4      <1>      sub     ah, al ; ah = i - nblines
3242                      <1>      ; else
3243                      <1>      ; vgamem_copy_pl4(cul,i,i-nblines,cols,nbcols,height);
3244 0000230D E88BFDFFFF <1>      call    vgamem_copy_pl4
3245                      <1> vga_g_down_pl1:
3246 00002312 86F5      <1>      xchg    dh, ch
3247 00002314 FECE      <1>      dec     dh
3248 00002316 38EE      <1>      cmp     dh, ch
3249 00002318 73E1      <1>      jnb     short vga_g_down_pl0
3250 0000231A C3        <1>      retn
3251                      <1>
3252                      <1> vga_g_down_pl2:
3253                      <1>      ; vgamem_fill_pl4(cul,i,cols,nbcols,height,attr);
3254 0000231B E8EBDFDFDF <1>      call    vgamem_fill_pl4
3255 00002320 EBF0      <1>      jmp     short vga_g_down_pl1
3256                      <1>
3257                      <1> ; 07/07/2016
3258                      <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
3259                      <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3260                      <1> ;-----
3261                      <1> ; SCROLL DOWN
3262                      <1> ; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
3263                      <1> ; ENTRY --
3264                      <1> ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
3265                      <1> ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
3266                      <1> ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
3267                      <1> ; BH = FILL VALUE FOR BLANKED LINES
3268                      <1> ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
3269                      <1> ; DS = DATA SEGMENT
3270                      <1> ; ES = REGEN SEGMENT
3271                      <1> ; EXIT --
3272                      <1> ; NOTHING, THE SCREEN IS SCROLLED
3273                      <1> ;-----
3274                      <1>
3275                      <1>      ; cl = upper left column
3276                      <1>      ; ch = upper left row
3277                      <1>      ; dl = lower righth column
3278                      <1>      ; dh = lower right row
3279                      <1>      ;
3280                      <1>      ; al = line count (AL=0 means blank entire fields)
3281                      <1>      ; bl = fill value for blanked lines
3282                      <1>      ; bh = unused
3283                      <1>
3284                      <1> GRAPHICS_DOWN:
3285                      <1>      ; 07/07/2016
3286                      <1>      ;AH = Current video mode, [CRT_MODE]
3287                      <1>      ;STD      ; SET DIRECTION
3288 00002322 80FC07      <1>      cmp     ah, 7
3289 00002325 0F87F4FEFFFF <1>      ja      vga_graphics_down
3290                      <1>      ;je     _n0
3291                      <1>
3292 0000232B 88C7        <1>      MOV     bh, al      ; save line count in BH
3293 0000232D 6689D0      <1>      MOV     AX, DX      ; GET LOWER RIGHT POSITION INTO AX REG
3294                      <1>
3295                      <1> ;-----      USE CHARACTER SUBROUTINE FOR POSITIONING
3296                      <1> ;-----      ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
3297                      <1>
3298 00002330 E8F1010000 <1>      CALL    GRAPH_POSN
3299 00002335 0FB7F8      <1>      MOVzx   eDI, AX      ; SAVE RESULT AS DESTINATION ADDRESS
3300                      <1>
3301                      <1> ;-----      DETERMINE SIZE OF WINDOW
3302                      <1>
3303 00002338 6629CA      <1>      SUB     DX, CX
3304 0000233B 6681C20101 <1>      ADD     DX, 101h      ; ADJUST VALUES
3305 00002340 C0E602      <1>      SAL     DH, 2        ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
3306                      <1>      ; AND EVEN/ODD ROWS
3307                      <1>
3308                      <1> ;-----      DETERMINE CRT MODE
3309                      <1>
3310 00002343 803D[225E0000]06 <1>      CMP     byte [CRT_MODE], 6 ; TEST FOR MEDIUM RES
3311 0000234A 7307        <1>      JNC     short _R12      ; FIND_SOURCE_DOWN
3312                      <1>
3313                      <1> ;-----      MEDIUM RES DOWN
3314 0000234C D0E2        <1>      SAL     DL, 1        ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
3315 0000234E 66D1E7      <1>      SAL     DI, 1        ; OFFSET *2 SINCE 2 BYTES/CHAR
3316 00002351 6647        <1>      INC     DI        ; POINT TO LAST BYTE
3317                      <1>
3318                      <1> ;-----      DETERMINE THE SOURCE ADDRESS IN THE BUFFER
3319                      <1>
3320                      <1> _R12:      ; FIND_SOURCE_DOWN
3321 00002353 81C700800B00 <1>      add     edi, 0B8000h
3322 00002359 6681C7F000 <1>      ADD     DI, 240      ; POINT TO LAST ROW OF PIXELS
3323 0000235E C0E702      <1>      sal     bh, 2        ; multiply number of lines by 4
3324 00002361 74(06)      <1>      JZ      short 6      ; IF ZERO, THEN BLANK ENTIRE FIELD
3325 00002363 B050        <1>      MOV     AL, 80      ; 80 BYTES/ROW
3326 00002365 F6E7        <1>      mul     bh      ; determine offset to source
3327 00002367 89FE        <1>      MOV     eSI, eDI      ; SET UP SOURCE
3328 00002369 6629C6      <1>      SUB     SI, AX      ; SUBTRACT THE OFFSET
3329 0000236C 88F4        <1>      MOV     AH, DH      ; NUMBER OF ROWS IN FIELD
3330 0000236E 28FC        <1>      sub     ah, bh      ; determine number to move
3331                      <1>
3332                      <1> ;-----      LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
3333                      <1>
3334                      <1> _R13:      ; ROW_LOOP_DOWN
3335 00002370 E823000000 <1>      CALL    _R17      ; MOVE ONE ROW
3336 00002375 6681EE5020 <1>      SUB     SI, 2000h+80 ; MOVE TO NEXT ROW
3337 0000237A 6681EF5020 <1>      SUB     DI, 2000h+80
3338 0000237F FECC        <1>      DEC     AH      ; NUMBER OF ROWS TO MOVE
3339 00002381 75ED        <1>      JNZ     short _R13      ; CONTINUE TILL ALL MOVED
3340                      <1>

```

```

3341 <1> ;----- FILL IN THE VACATED LINE(S)
3342 <1> _R14: ; CLEAR_ENTRY_DOWN
3343 00002383 88D8 <1> mov al, bl ; attribute to fill with
3344 <1> _R15_: ; CLEAR_LOOP_DOWN
3345 00002385 E82A000000 <1> CALL _R18 ; CLEAR A ROW
3346 0000238A 6681EF5020 <1> SUB DI, 2000h+80 ; POINT TO NEXT LINE
3347 0000238F FECF <1> dec bh ; number of lines to fill
3348 00002391 75F2 <1> JNZ short _R15_ ; CLEAR_LOOP_DOWN
3349 <1>
3350 00002393 C3 <1> retn ; EVERYTHING DONE
3351 <1>
3352 <1> _R16: ; BLANK_FIELD_DOWN
3353 00002394 88F7 <1> mov bh, dh ; set blank count to everything in field
3354 00002396 EBEB <1> JMP short _R14 ; CLEAR THE FIELD
3355 <1>
3356 <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
3357 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3358 <1>
3359 <1> ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
3360 <1>
3361 <1> _R17:
3362 00002398 0FB6CA <1> MOVzx ecx, DL ; NUMBER OF BYTES IN THE ROW
3363 0000239B 56 <1> PUSH eSI
3364 0000239C 57 <1> PUSH eDI ; SAVE POINTERS
3365 0000239D F3A4 <1> REP MOVSB ; MOVE THE EVEN FIELD
3366 0000239F 5F <1> POP eDI
3367 000023A0 5E <1> POP eSI
3368 000023A1 6681C60020 <1> ADD SI, 2000h
3369 000023A6 6681C70020 <1> ADD DI, 2000h ; POINT TO THE ODD FIELD
3370 000023AB 56 <1> PUSH eSI
3371 000023AC 57 <1> PUSH eDI ; SAVE THE POINTERS
3372 000023AD 88D1 <1> MOV CL, DL ; COUNT BACK
3373 000023AF F3A4 <1> REP MOVSB ; MOVE THE ODD FIELD
3374 000023B1 5F <1> POP eDI
3375 000023B2 5E <1> POP eSI ; POINTERS BACK
3376 000023B3 C3 <1> RETn ; RETURN TO CALLER
3377 <1>
3378 <1> ;----- CLEAR A SINGLE ROW
3379 <1>
3380 <1> _R18:
3381 000023B4 0FB6CA <1> MOVzx ecx, DL ; NUMBER OF BYTES IN FIELD
3382 000023B7 57 <1> PUSH eDI ; SAVE POINTER
3383 000023B8 F3AA <1> REP STOSB ; STORE THE NEW VALUE
3384 000023BA 5F <1> POP eDI ; POINTER BACK
3385 000023BB 6681C70020 <1> ADD DI, 2000h ; POINT TO ODD FIELD
3386 000023C0 57 <1> PUSH eDI
3387 000023C1 88D1 <1> MOV CL, DL
3388 000023C3 F3AA <1> REP STOSB ; FILL THE ODD FIELD
3389 000023C5 5F <1> POP eDI
3390 000023C6 C3 <1> RETn ; RETURN TO CALLER
3391 <1>
3392 <1> ; 04/07/2016
3393 <1> ; 01/07/2016
3394 <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
3395 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3396 <1> ;-----
3397 <1> ; GRAPHICS WRITE
3398 <1> ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
3399 <1> ; POSITION ON THE SCREEN.
3400 <1> ; ENTRY --
3401 <1> ; AL = CHARACTER TO WRITE
3402 <1> ; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
3403 <1> ; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
3404 <1> ; (0 IS USED FOR THE BACKGROUND COLOR)
3405 <1> ; CX = NUMBER OF CHARS TO WRITE
3406 <1> ; DS = DATA SEGMENT
3407 <1> ; ES = REGEN SEGMENT
3408 <1> ; EXIT --
3409 <1> ; NOTHING IS RETURNED
3410 <1> ;
3411 <1> ; GRAPHICS READ
3412 <1> ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
3413 <1> ; POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
3414 <1> ; CHARACTER GENERATOR CODE POINTS
3415 <1> ; ENTRY --
3416 <1> ; NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)
3417 <1> ; EXIT --
3418 <1> ; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
3419 <1> ;
3420 <1> ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
3421 <1> ; FOR THE 1ST 128 CHARS. TO ACCESS CHARS IN THE SECOND HALF, THE USER
3422 <1> ; MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO
3423 <1> ; POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
3424 <1> ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
3425 <1> ;-----
3426 <1>
3427 <1> GRAPHICS_WRITE:
3428 000023C7 25FF000000 <1> and eax, 0FFh ; ZERO TO HIGH OF CODE POINT
3429 000023CC 50 <1> PUSH eAX ; SAVE CODE POINT VALUE
3430 <1>
3431 <1> ;----- DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
3432 <1>
3433 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[A00C0100] <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[225E0000]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                ; CONVERT TO COLOR
3508 00002450 6621D8                 <1>      AND      AX, BX            ; SWAP HIGH/LOW BYTES FOR WORD MOVE
3509 00002453 86E0                     <1>      XCHG     AH, AL            ; AGAIN, IS THIS XOR FUNCTION
3510 00002455 F6C280                 <1>      TEST     DL, 80H            ; NO, JUST STORE THE VALUES
3511 00002458 7407                     <1>      JZ       short _S10          ; FUNCTION WITH FIRST HALF LOW
3512 0000245A 66338700200000         <1>      XOR      AX, [eDI+2000H]
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[225E0000]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 <1> MOV AL, [eSI] ; GET FIRST BYTE
3549 <1> MOV [eBP], AL ; SAVE IN STORAGE AREA
3550 <1> INC eBP ; NEXT LOCATION
3551 <1> MOV AL, [eSI+2000H] ; GET LOWER REGION BYTE
3552 <1> MOV [eBP], AL ; ADJUST AND STORE
3553 <1> INC eBP
3554 <1> ADD eSI, 80 ; POINTER INTO REGEN
3555 <1> DEC DH ; LOOP CONTROL
3556 <1> JNZ short S11 ; DO IT SOME MORE
3557 <1> JMP SHORT S14 ; GO MATCH THE SAVED CODE POINTS
3558 <1>
3559 <1> ;----- MEDIUM RESOLUTION READ
3560 <1> S12:
3561 <1> SAL SI, 1 ; OFFSET*2 SINCE 2 BYTES/CHAR
3562 <1> ;MOV DH, 4 ; NUMBER OF PASSES
3563 <1> S13:
3564 <1> CALL S23 ; GET BYTES FROM REGEN INTO SINGLE SAVE
3565 <1> ADD eSI, 2000H-2 ; GO TO LOWER REGION
3566 <1> CALL S23 ; GET THIS PAIR INTO SAVE
3567 <1> SUB eSI, 2000H-80+2 ; ADJUST POINTER BACK INTO UPPER
3568 <1> DEC DH
3569 <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 <1> MOV eDI, CRT_CHAR_GEN ; ESTABLISH ADDRESSING
3574 <1> SUB eBP, 8 ; ADJUST POINTER TO START OF SAVE AREA
3575 <1> MOV eSI, eBP
3576 <1> S15:
3577 <1> mov ax, 256 ; NUMBER TO TEST AGAINST
3578 <1> S16:
3579 <1> PUSH eSI ; SAVE SAVE AREA POINTER
3580 <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 <1> cmpsd ; compare first 4 bytes
3584 <1> jne short S17 ;
3585 <1> cmpsd ; compare last 4 bytes
3586 <1> S17:
3587 <1> POP eDI ; RECOVER THE POINTERS
3588 <1> POP eSI
3589 <1> ;JZ short S18 ; IF ZERO FLAG SET, THEN MATCH OCCURRED
3590 <1> je short S18
3591 <1> ;
3592 <1> ADD eDI, 8 ; NO MATCH, MOVE ON TO NEXT
3593 <1> dec ax ; NEXT CODE POINT
3594 <1> JNZ short S16 ; LOOP CONTROL
3595 <1> ; DO ALL OF THEM
3596 <1> ;----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
3597 <1> S18:
3598 <1> ADD eSP, 8 ; READJUST THE STACK, THROW AWAY SAVE
3599 <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 <1> PUSH CX ; SAVE REGISTER
3611 <1> ;MOV CX, 8 ; SHIFT COUNT REGISTER FOR ONE BYTE
3612 <1> mov cl, 8
3613 <1> S22:
3614 <1> ROR AL,1 ; SHIFT BITS, LOW BIT INTO CARRY FLAG
3615 <1> RCR BP,1 ; MOVE CARRY FLAG (LOW BIT INTO RESULTS
3616 <1> SAR BP,1 ; SIGN EXTEND HIGH BIT (DOUBLE IT)
3617 <1> ;LOOP S22 ; REPEAT FOR ALL 8 BITS
3618 <1> dec cl
3619 <1> jnz short S22
3620 <1> XCHG AX, BP ; MOVE RESULTS TO PARAMETER REGISTER
3621 <1> POP CX ; RECOVER REGISTER
3622 <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
3638 <1> ;-----
3639 <1> S23:
3640 <1> LODSW ; GET FIRST BYTE AND SECOND BYTES
3641 <1> XCHG AL, AH ; SWAP FOR COMPARE
3642 <1> MOV CX, 0C000H ; 2 BIT MASK TO TEST THE ENTRIES
3643 <1> MOV DL, 0 ; RESULT REGISTER
3644 <1> S24:
3645 <1> TEST AX, CX ; IS THIS SECTION BACKGROUND?
3646 <1> JZ short S25 ; IF ZERO, IT IS BACKGROUND (CARRY=0)

```



```

3647 00002511 F9          <1>      STC                      ; WASN'T, SO SET CARRY
3648                      <1> S25:
3649 00002512 D0D2        <1>      RCL     DL, 1              ; MOVE THAT BIT INTO THE RESULT
3650 00002514 66C1E902     <1>      SHR     CX, 2              ; MOVE THE MASK TO THE RIGHT BY 2 BITS
3651 00002518 73F2        <1>      JNC     short S24          ; DO IT AGAIN IF MASK DIDN'T FALL OUT
3652 0000251A 885500      <1>      MOV     [eBP], DL          ; STORE RESULT IN SAVE AREA
3653 0000251D 45          <1>      INC     eBP              ; ADJUST POINTER
3654 0000251E C3          <1>      RETn                     ; ALL DONE
3655                      <1>
3656                      <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
3657                      <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3658                      <1> ;-----
3659                      <1> ; V4_POSITION
3660                      <1> ; THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
3661                      <1> ; THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
3662                      <1> ; INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
3663                      <1> ; FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
3664                      <1> ; BE DOUBLED.
3665                      <1> ; ENTRY -- NO REGISTERS, MEMORY LOCATION @CURSOR_POSN IS USED
3666                      <1> ; EXIT--
3667                      <1> ; AX CONTAINS OFFSET INTO REGEN BUFFER
3668                      <1> ;-----
3669                      <1> S26:
3670 0000251F 0FB705[3E380100] <1>      movzx  eax, word [CURSOR_POSN]    ; GET CURRENT CURSOR
3671                      <1> GRAPH_POSN:
3672 00002526 53           <1>      PUSH   eBX              ; SAVE REGISTER
3673 00002527 0FB6D8      <1>      movzx  ebx, al              ; SAVE A COPY OF CURRENT CURSOR
3674 0000252A A0[245E0000] <1>      MOV     AL, [CRT_COLS]          ; GET BYTES PER COLUMN
3675 0000252F F6E4        <1>      MUL     AH              ; MULTIPLY BY ROWS
3676 00002531 66C1E002     <1>      SHL     AX, 2              ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
3677 00002535 01D8        <1>      ADD     eAX, eBX          ; DETERMINE OFFSET
3678 00002537 5B          <1>      POP     eBX              ; RECOVER POINTER
3679 00002538 C3          <1>      RETn                     ; ALL DONE
3680                      <1>
3681                      <1> ; 09/07/2016
3682                      <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
3683                      <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3684                      <1> ;-----
3685                      <1> ; SET_COLOR
3686                      <1> ; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
3687                      <1> ; AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
3688                      <1> ; INPUT
3689                      <1> ; (BH) HAS COLOR ID
3690                      <1> ; IF BH=0, THE BACKGROUND COLOR VALUE IS SET
3691                      <1> ; FROM THE LOW BITS OF BL (0-31)
3692                      <1> ; IF BH=1, THE PALETTE SELECTION IS MADE
3693                      <1> ; BASED ON THE LOW BIT OF BL:
3694                      <1> ; 0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
3695                      <1> ; 1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
3696                      <1> ; (BL) HAS THE COLOR VALUE TO BE USED
3697                      <1> ; OUTPUT
3698                      <1> ; THE COLOR SELECTION IS UPDATED
3699                      <1> ;-----
3700                      <1> SET_COLOR:
3701 00002539 803D[225E0000]07 <1>      cmp     byte [CRT_MODE], 7      ; 09/07/2016
3702 00002540 0F870EF0FFFF <1>      ja     VIDEO_RETURN          ; nothing to do for VGA modes
3703                      <1>
3704                      <1> ;MOV     DX, [ADDR_6845]          ; I/O PORT FOR PALETTE
3705                      <1> ;mov     dx, 3D4h
3706                      <1> ;ADD     DX,5              ; OVERSCAN PORT
3707 00002546 66BAD903     <1>      mov     dx, 3D9h
3708 0000254A A0[255E0000] <1>      MOV     AL, [CRT_PALETTE]      ; GET THE CURRENT PALETTE VALUE
3709 0000254F 08FF        <1>      OR      BH, BH              ; IS THIS COLOR 0?
3710 00002551 7512        <1>      JNZ     short M20          ; OUTPUT COLOR 1
3711                      <1>
3712                      <1> ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
3713                      <1>
3714 00002553 24E0        <1>      AND     AL, 0E0H          ; TURN OFF LOW 5 BITS OF CURRENT
3715 00002555 80E31F      <1>      AND     BL, 01FH          ; TURN OFF HIGH 3 BITS OF INPUT VALUE
3716 00002558 08D8        <1>      OR      AL, BL              ; PUT VALUE INTO REGISTER
3717                      <1> M19:
3718 0000255A EE          <1>      OUT     DX, AL              ; OUTPUT THE PALETTE
3719 0000255B A2[255E0000] <1>      MOV     [CRT_PALETTE], AL      ; OUTPUT COLOR SELECTION TO 3D9 PORT
3720 00002560 E9EFEFFFFF <1>      JMP     VIDEO_RETURN          ; SAVE THE COLOR VALUE
3721                      <1>
3722                      <1> ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
3723                      <1>
3724                      <1> M20:
3725 00002565 24DF        <1>      AND     AL, 0DFH          ; TURN OFF PALETTE SELECT BIT
3726 00002567 D0EB        <1>      SHR     BL, 1              ; TEST THE LOW ORDER BIT OF BL
3727 00002569 73EF        <1>      JNC     short M19          ; ALREADY DONE
3728 0000256B 0C20        <1>      OR      AL, 20H              ; TURN ON PALETTE SELECT BIT
3729 0000256D EBEB        <1>      JMP     short M19          ; GO DO IT
3730                      <1>
3731                      <1> ; 09/07/2016
3732                      <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
3733                      <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3734                      <1> ;-----
3735                      <1> ; READ DOT -- WRITE DOT
3736                      <1> ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
3737                      <1> ; DOT AT THE INDICATED LOCATION
3738                      <1> ; ENTRY --
3739                      <1> ; DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
3740                      <1> ; CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
3741                      <1> ; AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
3742                      <1> ; REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
3743                      <1> ; BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
3744                      <1> ; DS = DATA SEGMENT
3745                      <1> ; ES = REGEN SEGMENT
3746                      <1> ;
3747                      <1> ; EXIT
3748                      <1> ; AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY

```

```

3749 <1> ;-----
3750 <1>
3751 <1> READ_DOT:
3752 <1> ; 09/07/2016
3753 0000256F 8A25[225E0000] <1> mov ah, [CRT_MODE]
3754 00002575 80FC07 <1> cmp ah, 7 ; 6!?
3755 00002578 760A <1> jna short read_dot_cga
3756 <1>
3757 0000257A E8CB030000 <1> call vga_read_pixel
3758 <1> ; al = pixel value
3759 0000257F E9D5EFFFFF <1> jmp _video_return
3760 <1>
3761 <1> read_dot_cga:
3762 <1> ;je VIDEO_RETURN ; 7
3763 00002584 80FC04 <1> cmp ah, 4 ; graphics ?
3764 00002587 0F82C7EFFFFF <1> jb VIDEO_RETURN ; no, text mode, nothing to do
3765 <1>
3766 0000258D E855000000 <1> CALL R3 ; DETERMINE BYTE POSITION OF DOT
3767 00002592 8A06 <1> MOV AL, [eSI] ; GET THE BYTE
3768 00002594 20E0 <1> AND AL, AH ; MASK OFF THE OTHER BITS IN THE BYTE
3769 00002596 D2E0 <1> SHL AL, CL ; LEFT JUSTIFY THE VALUE
3770 00002598 88F1 <1> MOV CL, DH ; GET NUMBER OF BITS IN RESULT
3771 0000259A D2C0 <1> ROL AL, CL ; RIGHT JUSTIFY THE RESULT
3772 <1> ;JMP VIDEO_RETURN ; RETURN FROM VIDEO I/O
3773 0000259C 0FB6C0 <1> movzx eax, al
3774 0000259F E9B5EFFFFF <1> jmp _video_return
3775 <1>
3776 <1> ; 09/07/2016
3777 <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
3778 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3779 <1>
3780 <1> WRITE_DOT:
3781 <1> ; 09/07/2016
3782 000025A4 8A25[225E0000] <1> mov ah, [CRT_MODE]
3783 000025AA 80FC07 <1> cmp ah, 7 ; 6!?
3784 000025AD 760A <1> jna short write_dot_cga
3785 <1>
3786 000025AF E805030000 <1> call vga_write_pixel
3787 000025B4 E99BEFFFFF <1> jmp VIDEO_RETURN
3788 <1>
3789 <1> write_dot_cga:
3790 <1> ;je VIDEO_RETURN ; 7
3791 000025B9 80FC04 <1> cmp ah, 4 ; graphics ?
3792 000025BC 0F8292EFFFFF <1> jb VIDEO_RETURN ; no, text mode, nothing to do
3793 <1>
3794 <1> ;PUSH AX ; SAVE DOT VALUE
3795 000025C2 6650 <1> PUSH AX ; TWICE
3796 000025C4 E81E000000 <1> CALL R3 ; DETERMINE BYTE POSITION OF THE DOT
3797 000025C9 D2E8 <1> SHR AL, CL ; SHIFT TO SET UP THE BITS FOR OUTPUT
3798 000025CB 20E0 <1> AND AL, AH ; STRIP OFF THE OTHER BITS
3799 000025CD 8A0E <1> MOV CL, [eSI] ; GET THE CURRENT BYTE
3800 000025CF 665B <1> POP BX ; RECOVER XOR FLAG
3801 000025D1 F6C380 <1> TEST BL, 80H ; IS IT ON
3802 000025D4 750D <1> JNZ short R2 ; YES, XOR THE DOT
3803 000025D6 F6D4 <1> NOT AH ; SET MASK TO REMOVE THE INDICATED BITS
3804 000025D8 20E1 <1> AND CL, AH
3805 000025DA 08C8 <1> OR AL, CL ; OR IN THE NEW VALUE OF THOSE BITS
3806 <1> R1: ; FINISH_DOT
3807 000025DC 8806 <1> MOV [eSI], AL ; RESTORE THE BYTE IN MEMORY
3808 <1> ;POP AX
3809 000025DE E971EFFFFF <1> JMP VIDEO_RETURN ; RETURN FROM VIDEO I/O
3810 <1> R2: ; XOR_DOT
3811 000025E3 30C8 <1> XOR AL, CL ; EXCLUSIVE OR THE DOTS
3812 000025E5 EBF5 <1> JMP short R1 ; FINISH UP THE WRITING
3813 <1>
3814 <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
3815 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
3816 <1>
3817 <1> ;-----
3818 <1> ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
3819 <1> ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
3820 <1> ; ENTRY --
3821 <1> ; DX = ROW VALUE (0-199)
3822 <1> ; CX = COLUMN VALUE (0-639)
3823 <1> ; EXIT --
3824 <1> ; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
3825 <1> ; AH = MASK TO STRIP OFF THE BITS OF INTEREST
3826 <1> ; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
3827 <1> ; DH = # BITS IN RESULT
3828 <1> ; BX = MODIFIED
3829 <1> ;-----
3830 <1> R3:
3831 <1>
3832 <1> ;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
3833 <1> ;----- ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )
3834 <1>
3835 000025E7 0FB7F0 <1> movzx esi, ax ; WILL SAVE AL AND AH DURING OPERATION
3836 000025EA B028 <1> MOV AL, 40
3837 000025EC F6E2 <1> MUL DL ; AX= ADDRESS OF START OF INDICATED ROW
3838 000025EE A808 <1> TEST AL, 08H ; TEST FOR EVEN/ODD ROW CALCULATED
3839 000025F0 7404 <1> JZ short R4 ; JUMP IF EVEN ROW
3840 000025F2 6605D81F <1> ADD AX, 2000H-40 ; OFFSET TO LOCATION OF ODD ROWS ADJUST
3841 <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[225E0000]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[60070100] <1>      mov     esi, palette0
3915 00002654 EB15 <1>      jmp     short l_dac_p_4
3916 <1> l_dac_p_1:
3917 00002656 BE[20080100] <1>      mov     esi, palette1
3918 0000265B EB0E <1>      jmp     short l_dac_p_4
3919 <1> l_dac_p_2:
3920 0000265D BE[E0080100] <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[A0090100] <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
3944 <1> l_dac_p_6:
3945 00002680 EE <1>      out     dx, al ; outb(VGAREG_DAC_DATA,0);
3946 00002681 EE <1>      out     dx, al
3947 00002682 EE <1>      out     dx, al
3948 00002683 E2FB <1>      loop    l_dac_p_6
3949 <1> l_dac_p_7:
3950 <1>      ;pop esi
3951 00002685 C3 <1>      retn
3952 <1>

```

```

3953 <1> gray_scale_summing:
3954 <1> ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
3955 <1> ; (set_mode_vga)
3956 <1> ; derived from 'Plex86/Bochs VGABios' source code
3957 <1> ; vgabios-0.7a (2011)
3958 <1> ; by the LGPL VGABios developers Team (2001-2008)
3959 <1> ; 'vgabios.c', 'biosfn_perform_gray_scale_summing'
3960 <1> ;
3961 <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code
3962 <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
3963 <1> ;
3964 <1>
3965 <1> ; INPUT -> EBX = Start address (color index <= 255)
3966 <1> ; ECX = Count (<= 256)
3967 <1> ; OUTPUT -> (E)CX = 0
3968 <1> ; (Modifed registers: EAX, ECX, EDX, EBX)
3969 <1>
3970 00002686 66BADA03 <1> mov dx, 3DAh ; VGAREG_ACTL_RESET
3971 0000268A EC <1> in al, dx
3972 0000268B 30C0 <1> xor al, al ; 0
3973 0000268D 66BAC003 <1> mov dx, 3C0h ; VGAREG_ACTL_ADDRESS
3974 00002691 EE <1> out dx, al ; clear bit 5
3975 <1> ; (while loading palette registers)
3976 <1> ; set read address and switch to read mode
3977 <1> g_s_s_1:
3978 00002692 66BAC703 <1> mov dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
3979 00002696 88D8 <1> mov al, bl
3980 00002698 EE <1> out dx, al
3981 <1> ; get 6-bit wide RGB data values
3982 <1> ; intensity = (0.3*Red)+(0.59*Green)+(0.11*Blue)
3983 <1> ; i = ( ( 77*r + 151*g + 28*b ) + 0x80 ) >> 8;
3984 00002699 66BAC903 <1> mov dx, 3C9h ; VGAREG_DAC_DATA
3985 0000269D EC <1> in al, dx ; red
3986 0000269E B44D <1> mov ah, 77 ; 0.3* Red
3987 000026A0 F6E4 <1> mul ah
3988 000026A2 6650 <1> push ax
3989 000026A4 EC <1> in al, dx ; green
3990 000026A5 B497 <1> mov ah, 151 ; 0.59 * Green
3991 000026A7 F6E4 <1> mul ah
3992 000026A9 6650 <1> push ax
3993 000026AB EC <1> in al, dx ; blue
3994 000026AC B41C <1> mov ah, 28 ; 0.11 * Blue
3995 000026AE F6E4 <1> mul ah
3996 000026B0 665A <1> pop dx
3997 000026B2 6601D0 <1> add ax, dx
3998 000026B5 665A <1> pop dx
3999 000026B7 6601D0 <1> add ax, dx
4000 000026BA 66058000 <1> add ax, 80h
4001 000026BE B03F <1> mov al, 3Fh
4002 000026C0 38C4 <1> cmp ah, al
4003 000026C2 7602 <1> jna short g_s_s_2
4004 000026C4 88C4 <1> mov ah, al
4005 <1> g_s_s_2:
4006 000026C6 66BAC803 <1> mov dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
4007 000026CA 88D8 <1> mov al, bl ; color index
4008 000026CC EE <1> out dx, al
4009 000026CD 88E0 <1> mov al, ah ; intensity
4010 000026CF 6642 <1> inc dx ; 3C9h ; VGAREG_DAC_DATA
4011 000026D1 EE <1> out dx, al ; R (R=G=B)
4012 000026D2 88E0 <1> mov al, ah ; intensity
4013 000026D4 EE <1> out dx, al ; G (R=G=B)
4014 000026D5 88E0 <1> mov al, ah ; intensity
4015 000026D7 EE <1> out dx, al ; B (R=G=B)
4016 000026D8 6649 <1> dec cx
4017 000026DA 7404 <1> jz short g_s_s_3
4018 000026DC FEC3 <1> inc bl ; next color index value
4019 000026DE EBB2 <1> jmp short g_s_s_1
4020 <1> g_s_s_3:
4021 000026E0 66BADA03 <1> mov dx, 3DAh ; VGAREG_ACTL_RESET
4022 000026E4 EC <1> in al, dx
4023 000026E5 B020 <1> mov al, 20h
4024 000026E7 66BAC003 <1> mov dx, 3C0h ; VGAREG_ACTL_ADDRESS
4025 000026EB EE <1> out dx, al ; 20h -> set bit 5
4026 <1> ; (after loading palette regs)
4027 000026EC C3 <1> retn
4028 <1>
4029 <1> vga_write_char_attr:
4030 <1> vga_write_char_only:
4031 <1> ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
4032 <1> ;
4033 <1> ; derived from 'Plex86/Bochs VGABios' source code
4034 <1> ; vgabios-0.7a (2011)
4035 <1> ; by the LGPL VGABios developers Team (2001-2008)
4036 <1> ; 'vgabios.c', 'biosfn_write_char_attr'
4037 <1> ; 'biosfn_write_char_only'
4038 <1>
4039 <1> ; INPUT ->
4040 <1> ; [CRT_MODE] = current video mode (>7)
4041 <1> ; CX = Count of characters to write
4042 <1> ; AL = Character to write
4043 <1> ; BL = Color of character
4044 <1> ; OUTPUT ->
4045 <1> ; Regen buffer updated
4046 <1>
4047 000026ED 8A25[225E0000] <1> mov ah, [CRT_MODE]
4048 000026F3 668B15[3E380100] <1> mov dx, [CURSOR_POSN] ; cursor pos for page 0
4049 <1>
4050 000026FA BE[3E5E0000] <1> mov esi, vga_modes
4051 000026FF 89F7 <1> mov edi, esi
4052 00002701 83C710 <1> add edi, vga_mode_count
4053 <1> vga_wca_0:
4054 00002704 AC <1> lodsb

```



```

4055 00002705 38E0      <1>      cmp     al, ah ; [CRT_MODE]
4056 00002707 7405      <1>      je      short vga_wca_2
4057 00002709 39FE      <1>      cmp     esi, edi
4058 0000270B 72F7      <1>      jnb     short vga_wca_0
4059                                <1> vga_wca_1:
4060 0000270D C3          <1>      retn     ; nothing to do
4061                                <1> vga_wca_2:
4062 0000270E 83C64F      <1>      add     esi, vga_memmodel - (vga_modes + 1)
4063                                <1>      ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
4064                                <1>
4065                                <1>      ; biosfn_write_char_attr (car,page,attr,count)
4066                                <1>      ; AL = car, page = 0, BL = attr, CX = count
4067 00002711 803E04      <1>      cmp     byte [esi], PLANAR4
4068 00002714 741D      <1>      je      short vga_wca_planar
4069 00002716 803E03      <1>      cmp     byte [esi], PLANAR1
4070 00002719 7418      <1>      je      short vga_wca_planar
4071                                <1> vga_wca_linear8:
4072                                <1>      ; while((count-->0) && (xcurs<nbcols))
4073                                <1>      ; CX = count
4074 0000271B 6621C9      <1>      and     cx, cx
4075 0000271E 74ED      <1>      jz      short vga_wca_1
4076 00002720 3A15[245E0000] <1>      cmp     dl, [CRT_COLS]
4077 00002726 73E5      <1>      jnb     short vga_wca_1
4078                                <1>      ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols);
4079                                <1>      ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
4080                                <1>      ; [CRT_COLS] = nbcols
4081 00002728 E81E000000 <1>      call    write_gfx_char_lin
4082 0000272D 6649      <1>      dec     cx ; count
4083 0000272F FEC2      <1>      inc     dl ; xcurs
4084 00002731 EBE8      <1>      jmp     short vga_wca_linear8
4085                                <1> vga_wca_planar:
4086                                <1>      ; while((count-->0) && (xcurs<nbcols))
4087                                <1>      ; CX = count
4088 00002733 6621C9      <1>      and     cx, cx
4089 00002736 74D5      <1>      jz      short vga_wca_1
4090 00002738 3A15[245E0000] <1>      cmp     dl, [CRT_COLS]
4091 0000273E 73CD      <1>      jnb     short vga_wca_1
4092                                <1>      ; write_gfx_char_pl4(car,attr,xcurs,ycurs,nbcols,cheight);
4093                                <1>      ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
4094                                <1>      ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight
4095 00002740 E89D000000 <1>      call    write_gfx_char_pl4
4096 00002745 6649      <1>      dec     cx ; count
4097 00002747 FEC2      <1>      inc     dl ; xcurs
4098 00002749 EBE8      <1>      jmp     short vga_wca_planar
4099                                <1>
4100                                <1> write_gfx_char_lin:
4101                                <1>      ; 08/08/2016
4102                                <1>      ; 31/07/2016
4103                                <1>      ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
4104                                <1>      ;
4105                                <1>      ; derived from 'Plex86/Bochs VGABios' source code
4106                                <1>      ; vgabios-0.7a (2011)
4107                                <1>      ; by the LGPL VGABios developers Team (2001-2008)
4108                                <1>      ; 'vgabios.c', 'write_gfx_char_lin'
4109                                <1>
4110                                <1>      ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols)
4111                                <1>      ; INPUT ->
4112                                <1>      ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
4113                                <1>      ; [CRT_COLS] = nbcols
4114                                <1>      ; OUTPUT ->
4115                                <1>      ; Regen buffer updated
4116                                <1>
4117 0000274B 51          <1>      push    ecx
4118 0000274C 53          <1>      push    ebx
4119 0000274D 52          <1>      push    edx
4120 0000274E 50          <1>      push    eax
4121                                <1>      ; addr=xcurs*8+ycurs*nbcols*64;
4122                                <1>      ; 08/08/2016
4123 0000274F 0FB6F0      <1>      movzx   esi, al ; car
4124 00002752 0FB6C6      <1>      movzx   eax, dh ; ycurs
4125 00002755 8A25[245E0000] <1>      mov     ah, [CRT_COLS] ; nbcols
4126 0000275B F6E4      <1>      mul     ah
4127                                <1>      ;shl     ax, 6 ; * 64
4128 0000275D 66C1E003 <1>      shl     ax, 3 ; * 8
4129                                <1>      ;sub     dh, dh
4130                                <1>      ;shl     dx, 3 ; xcurs * 8
4131                                <1>      ;movzx   edi, dx
4132 00002761 0FB6FA      <1>      movzx   edi, dl
4133 00002764 66C1E703 <1>      shl     di, 3 ; xcurs * 8
4134 00002768 30F6      <1>      xor     dh, dh
4135 0000276A 8A15[265E0000] <1>      mov     dl, [CHAR_HEIGHT]
4136 00002770 66F7E2      <1>      mul     dx
4137                                <1>      ; eax = ycurs*nbcols*8*[CHAR_HEIGHT]
4138 00002773 01C7      <1>      add     edi, eax ; addr
4139 00002775 81C700000A00 <1>      add     edi, 0A0000h
4140                                <1>      ;shl     si, 3 ; car * 8
4141 0000277B 30E4      <1>      xor     ah, ah
4142 0000277D A0[265E0000] <1>      mov     al, [CHAR_HEIGHT]
4143 00002782 66F7E6      <1>      mul     si
4144 00002785 6689C6      <1>      mov     si, ax
4145                                <1>      ; esi = src = car * 8
4146                                <1>      ; esi = src = car * [CHAR_HEIGHT]
4147                                <1>      ; i = 0
4148                                <1>      ;add     esi, vgafont8 ; fdata [src+i]
4149                                <1>      ; 08/08/2016
4150 00002788 A1[CE450100] <1>      mov     eax, [VGA_INT43H]
4151 0000278D 3D[A0220100] <1>      cmp     eax, vgafont16
4152 00002792 740F      <1>      je      short wgfxl_0
4153 00002794 3D[A0140100] <1>      cmp     eax, vgafont14
4154 00002799 7408      <1>      je      short wgfxl_0
4155 0000279B 81C6[A00C0100] <1>      add     esi, vgafont8
4156 000027A1 EB02      <1>      jmp     short wgfxl_1

```



```

4157
4158 000027A3 01C6
4159
4160 000027A5 28FF
4161
4162
4163 000027A7 57
4164 000027A8 0FB605[245E0000]
4165 000027AF F6E7
4166 000027B1 66C1E003
4167
4168 000027B5 01C7
4169 000027B7 B180
4170
4171
4172 000027B9 29D2
4173
4174 000027BB 8A06
4175 000027BD 20C8
4176 000027BF 7402
4177 000027C1 88D8
4178
4179
4180 000027C3 AA
4181
4182
4183 000027C4 80FA07
4184 000027C7 720E
4185 000027C9 5F
4186
4187
4188
4189 000027CA FEC7
4190 000027CC 3A3D[265E0000]
4191 000027D2 7309
4192 000027D4 46
4193 000027D5 EBD0
4194
4195 000027D7 D0E9
4196 000027D9 FEC2
4197 000027DB EBDE
4198
4199 000027DD 58
4200 000027DE 5A
4201 000027DF 5B
4202 000027E0 59
4203 000027E1 C3
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221 000027E2 51
4222 000027E3 53
4223 000027E4 52
4224 000027E5 50
4225
4226
4227 000027E6 8A25[265E0000]
4228 000027EC 80FC10
4229 000027EF 7507
4230
4231 000027F1 BE[A0220100]
4232 000027F6 EB13
4233
4234 000027F8 80FC0E
4235 000027FB 7507
4236 000027FD BE[A0140100]
4237 00002802 EB07
4238
4239
4240
4241 00002804 BE[A00C0100]
4242 00002809 B408
4243
4244
4245 0000280B F6E4
4246 0000280D 25FFFF0000
4247
4248 00002812 01C6
4249
4250 00002814 88F0
4251 00002816 8A25[245E0000]
4252 0000281C F6E4
4253
4254
4255 0000281E 66C1E003
4256
4257
4258

<1> wgfxl_0:
<1>     add     esi, eax
<1> wgfxl_1:
<1>     sub     bh, bh ; i = 0
<1> wgfxl_2:
<1>     ; for(i=0;i<8;i++)
<1>     push    edi ; addr
<1>     movzx   eax, byte [CRT_COLS] ; ncols
<1>     mul     bh ; ncols*i
<1>     shl     ax, 3 ; i*ncols*8
<1>     ; dest=addr+i*ncols*8;
<1>     add     edi, eax ; dest + j ; j = 0
<1>     mov     cl, 80h ; mask = 0x80;
<1>     ; esi = fdata + src + i
<1>     ; for(j=0;j<8;j++)
<1>     sub     edx, edx ; j = 0
<1> wgfxl_3:
<1>     mov     al, [esi] ; al = fdata[src+i]
<1>     and     al, cl ; if (fdata[src+i] & mask)
<1>     jz      short wgfxl_4 ; data = 0, zf = 1
<1>     mov     al, bl ; data = attr;
<1> wgfxl_4:
<1>     ; write_byte(0xa000,dest+j,data);
<1>     stosb   ; dest + j (+ 0A0000h)
<1>     ;inc    dl ; j++
<1>     ;cmp    dl, 8
<1>     cmp     dl, 7
<1>     jnb     short wgfxl_5
<1>     pop     edi
<1>     ; 08/08/2016
<1>     ;cmp    bh, 7
<1>     ;jnb    short wgfxl_6
<1>     inc     bh ; i++
<1>     cmp     bh, [CHAR_HEIGHT]
<1>     jnb     short wgfxl_6
<1>     inc     esi
<1>     jmp     short wgfxl_2
<1> wgfxl_5:
<1>     shr     cl, 1 ; mask >= 1;
<1>     inc     dl ; j++
<1>     jmp     short wgfxl_3
<1> wgfxl_6:
<1>     pop     eax
<1>     pop     edx
<1>     pop     ebx
<1>     pop     ecx
<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,ncols,cheight)
<1>     ; INPUT ->
<1>     ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
<1>     ; [CRT_COLS] = ncols, [CHAR_HEIGHT] = cheight
<1>     ; OUTPUT ->
<1>     ; Regen buffer updated
<1>
<1>     push    ecx
<1>     push    ebx
<1>     push    edx
<1>     push    eax
<1> wgfxpl_f0:
<1>     ; switch(cheight)
<1>     mov     ah, [CHAR_HEIGHT]
<1>     cmp     ah, 16 ; case 16:
<1>     jne     short wgfxpl_f1
<1>     ; fdata = &vgafont16;
<1>     mov     esi, vgafont16
<1>     jmp     short wgfxpl_f3
<1> wgfxpl_f1:
<1>     cmp     ah, 14 ; case 14:
<1>     jne     short wgfxpl_f2
<1>     mov     esi, vgafont14
<1>     jmp     short wgfxpl_f3
<1> wgfxpl_f2:
<1>     ; default:
<1>     ; fdata = &vgafont8;
<1>     mov     esi, vgafont8
<1>     mov     ah, 8
<1> wgfxpl_f3:
<1>     ; al = car
<1>     mul     ah ; ah = cheight
<1>     and     eax, 0FFFFh ; car * cheight
<1>     ; src = car * cheight;
<1>     add     esi, eax ; esi = fdata[src+i]
<1>     ; addr=xcurs*8+ycurs*ncols*64;
<1>     mov     al, dh ; ycurs
<1>     mov     ah, [CRT_COLS] ; ncols
<1>     mul     ah
<1>     ; 08/08/2016
<1>     ;shl    ax, 6 ; * 64
<1>     shl     ax, 3 ; * 8
<1>     ;sub    dh, dh ; 0
<1>     ;shl    dx, 3 ; xcurs * 8
<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[265E0000]    <1>      mov  dl, [CHAR_HEIGHT]
4263 00002831 66F7E2      <1>      mul  dx
4264      <1>      ; eax = ycurs*nbcolls*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[245E0000] <1>      movzx eax, byte [CRT_COLS] ; nbcolls
4294 0000286F F6E7      <1>      mul  bh ; nbcolls*i
4295      <1>      ; dest=addr+i*nbcolls
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>      jnb  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[265E0000] <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[3E5E0000] <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>      jnb     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[245E0000] <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[245E0000] <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>
4454          <1>      ; INPUT ->
4455          <1>      ;      DX = row (0-239)
4456          <1>      ;      CX = column (0-799)
4457          <1>      ;      (AH = [CRT_MODE])
4458          <1>      ; OUTPUT ->
4459          <1>      ;      AL = pixel value
4460          <1>
4461          <1>      ;mov     ah, [CRT_MODE]
4462 0000294A BE[3E5E0000] <1>      mov     esi, vga_modes

```

4463	0000294F	89F7	<1>	mov	edi, esi
4464	00002951	83C710	<1>	add	edi, vga_mode_count
4465			<1>	vga_rp_0:	
4466	00002954	AC	<1>	lods	sb
4467	00002955	38E0	<1>	cmp	al, ah ; [CRT_MODE]
4468	00002957	7405	<1>	je	short vga_rp_1
4469	00002959	39FE	<1>	cmp	esi, edi
4470	0000295B	72F7	<1>	jb	short vga_rp_0
4471	0000295D	C3	<1>	retn	; nothing to do
4472			<1>	vga_rp_1:	
4473	0000295E	83C64F	<1>	add	esi, vga_memmodel - (vga_modes + 1)
4474			<1>		; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
4475	00002961	BF00000A00	<1>	mov	edi, 0A0000h
4476			<1>		;
4477	00002966	803E04	<1>	cmp	byte [esi], PLANAR4
4478	00002969	741D	<1>	je	short vga_rp_planar
4479	0000296B	803E03	<1>	cmp	byte [esi], PLANAR1
4480	0000296E	7418	<1>	je	short vga_rp_planar
4481			<1>	vga_rp_linear8:	
4482			<1>		; addr=CX+DX*(read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS)*8);
4483	00002970	0FB605[245E0000]	<1>	movzx	eax, byte [CRT_COLS] ; = [VGA_COLS] ; nbcols
4484	00002977	66C1E003	<1>	shl	ax, 3 ; * 8
4485	0000297B	66F7E2	<1>	mul	dx
4486	0000297E	50	<1>	push	eax
4487			<1>		;mov edi, 0A0000h
4488	0000297F	6601CF	<1>	add	di, cx
4489	00002982	58	<1>	pop	eax
4490	00002983	01C7	<1>	add	edi, eax ; addr
4491			<1>		; attr=read_byte(0xa000,addr);
4492	00002985	8A07	<1>	mov	al, [edi] ; pixel value
4493	00002987	C3	<1>	retn	
4494			<1>	vga_rp_planar:	
4495			<1>		; addr = CX/8+DX*read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS);
4496	00002988	0FB7C1	<1>	movzx	eax, cx
4497	0000298B	66C1E803	<1>	shr	ax, 3 ; CX/8
4498	0000298F	50	<1>	push	eax
4499	00002990	28E4	<1>	sub	ah, ah ; 0
4500	00002992	A0[245E0000]	<1>	mov	al, [CRT_COLS] ; = [VGA_COLS] ; nbcols
4501	00002997	66F7E2	<1>	mul	dx
4502			<1>		;mov edi, 0A0000h
4503	0000299A	6601C7	<1>	add	di, ax
4504	0000299D	58	<1>	pop	eax
4505	0000299E	01C7	<1>	add	edi, eax ; addr
4506	000029A0	80E107	<1>	and	cl, 7
4507	000029A3	B580	<1>	mov	ch, 80h ; mask
4508	000029A5	D2ED	<1>	shr	ch, cl ; mask = 0x80 >> (CX & 0x07);
4509			<1>		; attr = 0x00;
4510	000029A7	30DB	<1>	xor	bl, bl ; attr = bl = 0,
4511	000029A9	30C9	<1>	xor	cl, cl ; i = cl = 0
4512			<1>		; for(i=0;i<4;i++)
4513			<1>		{
4514			<1>		; outw(VGAREG_GRDC_ADDRESS, (i << 8) 0x04);
4515			<1>		; data = read_byte(0xa000,addr) & mask;
4516			<1>		; if (data > 0) attr = (0x01 << i);
4517			<1>		;
4518			<1>	vga_rp_2:	
4519	000029AB	88CC	<1>	mov	ah, cl ; i << 8
4520	000029AD	B004	<1>	mov	al, 4 ; 0x04
4521	000029AF	66BACE03	<1>	mov	dx, 3CEh ; VGAREG_GRDC_ADDRESS
4522	000029B3	66EF	<1>	out	dx, ax
4523			<1>		; data = read_byte(0xa000,addr) & mask;
4524	000029B5	8A07	<1>	mov	al, [edi]
4525	000029B7	20E8	<1>	and	al, ch ; & mask
4526			<1>		; if (data > 0) attr = (0x01 << i);
4527	000029B9	08C0	<1>	or	al, al
4528	000029BB	7408	<1>	jz	short vga_rp_3 ; al = 0
4529	000029BD	B701	<1>	mov	bh, 1
4530	000029BF	D2E7	<1>	shl	bh, cl ; (0x01 << i)
4531	000029C1	08FB	<1>		


```

4565      <1>      ; attr = color (BL)
4566      <1>      ; 'flag' not used
4567      <1>
4568 000029CC 8A25[225E0000] <1>      mov     ah, [CRT_MODE]
4569 000029D2 88C7          <1>      mov     bh, al ; character
4570 000029D4 668B15[3E380100] <1>      mov     dx, [CURSOR_POSN] ; cursor pos for page 0
4571      <1>
4572 000029DB BE[465E0000] <1>      mov     esi, vga_g_modes
4573 000029E0 89F7          <1>      mov     edi, esi
4574 000029E2 83C708        <1>      add     edi, vga_g_mode_count
4575      <1> vga_wtty_0:
4576 000029E5 AC          <1>      lodsb
4577 000029E6 38E0          <1>      cmp     al, ah ; [CRT_MODE]
4578 000029E8 7405          <1>      je      short vga_wtty_2
4579 000029EA 39FE          <1>      cmp     esi, edi
4580 000029EC 72F7          <1>      jnb     short vga_wtty_0
4581      <1> vga_wtty_1:
4582 000029EE C3          <1>      retn    ; nothing to do
4583      <1> vga_wtty_2:
4584 000029EF 80FF07        <1>      cmp     bh, 07h ; bell (beep)
4585 000029F2 74D2          <1>      je      short vga_beeper ; ull
4586 000029F4 80FF08        <1>      cmp     bh, 08h ; backspace
4587 000029F7 7508          <1>      jne     short vga_wtty_3
4588      <1>      ; if(xcurs>0)xcurs--;
4589 000029F9 08D2          <1>      or      dl, dl ; xcurs (column)
4590 000029FB 74F1          <1>      jz      short vga_wtty_1
4591 000029FD FECA          <1>      dec     dl ; xcurs--;
4592 000029FF EB59          <1>      jmp     short vga_wtty_12
4593      <1> vga_wtty_3:
4594 00002A01 80FF0D        <1>      cmp     bh, 0Dh ; carriage return (\r)
4595 00002A04 7504          <1>      jne     short vga_wtty_4
4596      <1>      ; xcurs=0;
4597 00002A06 28D2          <1>      sub     dl, dl ; 0
4598 00002A08 EB50          <1>      jmp     short vga_wtty_12
4599      <1> vga_wtty_4:
4600 00002A0A 80FF0A        <1>      cmp     bh, 0Ah ; new line (\n)
4601 00002A0D 7504          <1>      jne     short vga_wtty_5
4602      <1>      ; ycurs++;
4603 00002A0F FEC6          <1>      inc     dh ; next row
4604 00002A11 EB62          <1>      jmp     short vga_wtty_11
4605      <1> vga_wtty_5:
4606 00002A13 80FF09        <1>      cmp     bh, 09h ; tab stop
4607 00002A16 7527          <1>      jne     short vga_wtty_8
4608 00002A18 88D0          <1>      mov     al, dl
4609 00002A1A 6698          <1>      cbw
4610 00002A1C B108          <1>      mov     cl, 8
4611 00002A1E F6F1          <1>      div     cl
4612 00002A20 28E1          <1>      sub     cl, ah
4613      <1>      ;
4614 00002A22 B720          <1>      mov     bh, 20h ; space
4615      <1> vga_wtty_6: ; tab stop loop
4616 00002A24 6651          <1>      push    cx
4617 00002A26 6653          <1>      push    bx
4618 00002A28 E812000000    <1>      call    vga_wtty_8
4619 00002A2D 665B          <1>      pop     bx ; bh = character, bl = color
4620 00002A2F 6659          <1>      pop     cx
4621 00002A31 FEC9          <1>      dec     cl
4622 00002A33 7409          <1>      jz      short vga_wtty_7
4623 00002A35 668B15[3E380100] <1>      mov     dx, [CURSOR_POSN] ; new cursor position (pg 0)
4624 00002A3C EBE6          <1>      jmp     short vga_wtty_6
4625      <1> vga_wtty_7:
4626 00002A3E C3          <1>      retn
4627      <1>      ;
4628      <1> vga_wtty_8:
4629 00002A3F 83C64F        <1>      add     esi, vga_g_memmodel - (vga_g_modes + 1)
4630      <1>      ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
4631 00002A42 BF00000A00    <1>      mov     edi, 0A0000h
4632      <1>      ;
4633 00002A47 88F8          <1>      mov     al, bh ; character
4634      <1>      ;
4635 00002A49 803E04        <1>      cmp     byte [esi], PLANAR4
4636 00002A4C 7414          <1>      je      short vga_wtty_planar
4637 00002A4E 803E03        <1>      cmp     byte [esi], PLANAR1
4638 00002A51 740F          <1>      je      short vga_wtty_planar
4639      <1> vga_wtty_linear8:
4640      <1>      ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols);
4641      <1>      ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,
4642      <1>      ; [CRT_COLS] = nbcols
4643 00002A53 E8F3FCFFFF    <1>      call    write_gfx_char_lin
4644 00002A58 EB0D          <1>      jmp     short vga_wtty_9
4645      <1>
4646      <1> vga_wtty_12:
4647      <1>      ; 09/07/2016
4648      <1>      ; set cursor position
4649      <1>      ; NOTE: Hardware cursor position will not be set
4650      <1>      ; in any VGA modes (>7)
4651      <1>      ; But, cursor position will be saved into
4652      <1>      ; [CURSOR_POSN].
4653      <1>      ; TRDOS 386 (TRDOS v2.0) uses only one page
4654      <1>      ; (page 0) for all graphics modes.
4655      <1>
4656 00002A5A 668915[3E380100] <1>      mov     [CURSOR_POSN], dx ; save cursor pos for pg 0
4657      <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,height);
4664      <1>      ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,
4665      <1>      ; [CRT_COLS]= nbcols, [CHAR_HEIGHT] = height
4666 00002A62 E87BFDFFFF    <1>      call    write_gfx_char_pl4

```



```

4667
4668 00002A67 FEC2
4669
4670
4671
4672 00002A69 3A15[245E0000]
4673 00002A6F 7204
4674 00002A71 28D2
4675 00002A73 FEC6
4676
4677
4678
4679 00002A75 3A35[2A5E0000]
4680 00002A7B 72DD
4681
4682
4683
4684
4685
4686
4687 00002A7D B001
4688 00002A7F 28DB
4689 00002A81 6629C9
4690
4691
4692 00002A84 8A35[2A5E0000]
4693 00002A8A FECE
4694
4695 00002A8C 6652
4696 00002A8E 8A15[245E0000]
4697 00002A94 FECA
4698
4699 00002A96 8A25[225E0000]
4700
4701
4702 00002A9C E808F5FFFF
4703
4704 00002AA1 665A
4705
4706 00002AA3 EBB5
4707
4708
4709
4710
4711
4712
4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735 00002AA5 08C0
4736 00002AA7 7404
4737 00002AA9 3C10
4738 00002AAB 7511
4739
4740 00002AAD E8B7000000
4741 00002AB2 721C
4742 00002AB4 E8C2000000
4743 00002AB9 E996EAFFFF
4744
4745
4746
4747
4748
4749
4750 00002ABE 3C01
4751 00002AC0 7404
4752 00002AC2 3C11
4753 00002AC4 7511
4754
4755
4756
4757 00002AC6 E8EE010000
4758 00002ACB E984EAFFFF
4759
4760 00002AD0 29C0
4761 00002AD2 E982EAFFFF
4762
4763
4764
4765
4766
4767
4768 00002AD7 3C02

<1> vga_wtty_9:
<1>     inc    dl ; xcurs++;
<1> vga_wtty_10:
<1>     ; Do we need to wrap ?
<1>     ; if(xcurs==nbcols)
<1>     cmp     dl, [CRT_COLS] ; [VGA_COLS]
<1>     jb      short vga_wtty_11 ; no
<1>     sub     dl, dl ; xcurs=0;
<1>     inc     dh ; ycurs++;
<1> vga_wtty_11:
<1>     ; Do we need to scroll ?
<1>     ; if(ycurs==nbrows)
<1>     cmp     dh, [VGA_ROWS]
<1>     jb      short vga_wtty_12 ; no
<1>     ;
<1>     ; biosfn_scroll (nblines,attr,rul,cul,rlr,clr,page,dir)
<1>     ; al = nblines = 1, bl = attr (color) = 0
<1>     ; ch = rul, cl = cul, dh = rlr, dl = clr, page = 0
<1>     ; dir = SCROLL_UP
<1>
<1>     mov     al, 1
<1>     sub     bl, bl ; 0 ; blank/black line (attr=0) will be used
<1>     sub     cx, cx ; 0,0
<1>
<1>     ; 06/08/2016
<1>     mov     dh, [VGA_ROWS]
<1>     dec     dh ; nbrows -1
<1>
<1>     push    dx      ; 04/08/2016
<1>     mov     dl, [CRT_COLS]
<1>     dec     dl ; nbcols -1
<1>
<1>     mov     ah, [CRT_MODE]
<1>
<1>     ; biosfn_scroll(0x01,0x00,0,0,nbrows-1,nbcols-1,page,SCROLL_UP);
<1>     call    vga_graphics_up
<1>     ; 04/08/2016
<1>     pop     dx
<1>     ;dec    dh ; ycurs-=1
<1>     jmp     short vga_wtty_12
<1>
<1> font_setup:
<1>     ; 09/07/2016
<1>     ; character generator (font loading) functions
<1>     ;
<1>     ; derived from 'Plex86/Bochs VGABios' source code
<1>     ; vgabios-0.7a (2011)
<1>     ; by the LGPL VGABios developers Team (2001-2008)
<1>     ; 'vgabios.c', 'intl0_func'
<1>
<1>     ; AX = 1100H ; Load User-Defined Font (EGA/VGA)
<1>     ;
<1>     ; BH      height of each character (bytes per character definition)
<1>     ; (BL      font block to load (EGA: 0-3; VGA: 0-7))
<1>     ; CX      number of characters to redefine (<=256)
<1>     ; DX      ASCII code of the first character defined at ES:BP
<1>     ; EBP      address of font-definition information
<1>     ;          (in user's memory space)
<1>
<1>     ; case 0x11:
<1>     ; switch(GET_AL())
<1>     ; {
<1>     ; case 0x00:
<1>     ;     ; case 0x10:
<1>     ;     biosfn_load_text_user_pat(GET_AL(),ES,BP,CX,DX,GET_BL(),GET_BH());
<1>     ;     break;
<1>
<1>     ; AX = 1110H ; Load and Activate User-Defined Font (EGA/VGA)
<1>     or      al, al ; 0
<1>     jz      short font_setup_0
<1>     cmp     al, 10h
<1>     jne     short font_setup_1
<1> font_setup_0:
<1>     call    transfer_user_fonts
<1>     jc      short font_setup_error
<1>     call    load_text_user_pat
<1>     jmp     VIDEO_RETURN
<1> font_setup_1:
<1>     ; AX = 1101H ; Load ROM 8x14 Character Set (EGA/VGA)
<1>     ; case 0x01:
<1>     ;     ; case 0x11:
<1>     ;     biosfn_load_text_8_14_pat(GET_AL(),GET_BL());
<1>     ;     break;
<1>     cmp     al, 1
<1>     je      short font_setup_2
<1>     cmp     al, 11h
<1>     jne     short font_setup_3
<1> font_setup_2:
<1>     ; AX = 1111H ; Load and Activate ROM 8x14 Character Set (EGA/VGA)
<1>     ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
<1>     call    load_text_8_14_pat
<1>     jmp     VIDEO_RETURN
<1> font_setup_error:
<1>     sub     eax, eax ; 0 -> fonts could not be loaded
<1>     jmp     _video_return
<1> font_setup_3:
<1>     ; AX = 1102H ; Load ROM 8x8 Character Set (EGA/VGA)
<1>     ; case 0x02:
<1>     ;     ; case 0x12:
<1>     ;     biosfn_load_text_8_8_pat(GET_AL(),GET_BL());
<1>     ;     break;
<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 E8A5BC0000                 <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                                     <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>      jnb     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
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
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 FECF        <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[3B5E0000] <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[265E0000] <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[2A5E0000] <1>      mov     [VGA_ROWS], al ; rows (not 'rows-1' !)
5096          <1>      ; write_word(BIOSMEM_SEG, BIOSMEM_PAGE_SIZE, rows * cols * 2);
5097 00002CA7 8A25[245E0000] <1>      mov     ah, [CRT_COLS]
5098 00002CAD F6E4    <1>      mul     ah
5099 00002CAF 66D1E0  <1>      shl     ax, 1
5100 00002CB2 66A3[BC450100] <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[A0140100] <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
5168          <1>      ;
5169 00002CDE 58      <1>      pop     eax
5170          <1>      ; if(AL>=0x10)
5171 00002CDF 3C10    <1>      cmp     al, 10h
5172 00002CE1 7205    <1>      jb     short lt8_14_4
5173          <1>      ; BH = 14
5174          <1>      ; set_scan_lines(14);
5175 00002CE3 E849FFFFFF <1>      call    set_scan_lines
5176          <1> lt8_14_4:

```



```

5177 00002CE8 C3      <1>      retn
5178                  <1>
5179                  <1> load_text_8_8_pat:
5180                  <1>      ; 26/07/2016
5181                  <1>      ; 25/07/2016
5182                  <1>      ; 23/07/2016
5183                  <1>      ; 09/07/2016
5184                  <1>      ; load user defined (EGA/VGA) text fonts
5185                  <1>      ;
5186                  <1>      ; derived from 'Plex86/Bochs VGABios' source code
5187                  <1>      ; vgabios-0.7a (2011)
5188                  <1>      ; by the LGPL VGABios developers Team (2001-2008)
5189                  <1>      ; 'vgabios.c', 'biosfn_load_text_8_8_pat'
5190                  <1>
5191                  <1>      ; biosfn_load_text_8_8_pat (AL,BL)
5192                  <1>
5193                  <1>      ; get_font_access();
5194                  <1>      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
5195                  <1>      ; for(i=0;i<0x100;i++)
5196                  <1>      ; {
5197                  <1>      ;     src = i * 8;
5198                  <1>      ;     dest = blockaddr + i * 32;
5199                  <1>      ;     memcpyb(0xA000, dest, 0xC000, vgafont8+src, 8);
5200                  <1>      ; }
5201                  <1>      ; release_font_access();
5202                  <1>      ; if(AL>=0x10)
5203                  <1>      ; {
5204                  <1>      ;     set_scan_lines(8);
5205                  <1>      ; }
5206                  <1>
5207 00002CE9 50      <1>      push    eax
5208 00002CEA E8CEFEFFFF <1>      call   get_font_access
5209                  <1>
5210                  <1>      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
5211                  <1>      ;mov    dl, bl
5212                  <1>      ;and    dl, 3
5213                  <1>      ;shl    dx, 14
5214                  <1>      ;xchg   dx, bx
5215                  <1>      ;and    dl, 4
5216                  <1>      ;shl    dx, 11
5217                  <1>      ;add    dx, bx
5218                  <1>
5219                  <1>      ;xor    dx, dx ; blockaddr = 0
5220                  <1>      ; Always block 0 for TRDOS 386 ! (blockaddr=0(
5221                  <1>
5222 00002CEF 28DB      <1>      sub     bl, bl ; i = 0
5223 00002CF1 B708      <1>      mov     bh, 8
5224 00002CF3 BE[A00C0100] <1>      mov     esi, vgafont8
5225 00002CF8 BF00000A00 <1>      mov     edi, 0A0000h
5226                  <1> lt8_8_1:
5227                  <1>      ;mov    al, bl
5228                  <1>      ;mul    bh
5229                  <1>      ;movzx  esi, ax
5230                  <1>      ;add    esi, vgafont8
5231                  <1>      ;mov    al, bl
5232                  <1>      ;sub    ah, ah
5233                  <1>      ;shl    ax, 5 ; * 32
5234                  <1>      ;;add   ax, dx ; blockaddr + i * 32;
5235                  <1>      ;movzx  edi, ax ; dest
5236                  <1>      ;add    edi, 0A0000h
5237 00002CFD 0FB6CF      <1>      movzx   ecx, bh
5238 00002D00 F3A4      <1>      rep     movsb
5239 00002D02 83C718      <1>      add     edi, 24 ; 32 - 8
5240 00002D05 FEC3      <1>      inc     bl
5241 00002D07 75F4      <1>      jnz     short lt8_8_1
5242                  <1>      ;
5243 00002D09 E8E4FEFFFF <1>      call   release_font_access
5244                  <1>      ;
5245 00002D0E 58      <1>      pop     eax
5246                  <1>      ; if(AL>=0x10)
5247 00002D0F 3C10      <1>      cmp     al, 10h
5248 00002D11 7205      <1>      jb      short lt8_8_2
5249                  <1>      ; BH = 8
5250                  <1>      ; set_scan_lines(8);
5251 00002D13 E819FFFFFF <1>      call   set_scan_lines
5252                  <1> lt8_8_2:
5253 00002D18 C3      <1>      retn
5254                  <1>
5255                  <1> load_text_8_16_pat:
5256                  <1>      ; 26/07/2016
5257                  <1>      ; 25/07/2016
5258                  <1>      ; 23/07/2016
5259                  <1>      ; 09/07/2016
5260                  <1>      ; load user defined (EGA/VGA) text fonts
5261                  <1>      ;
5262                  <1>      ; derived from 'Plex86/Bochs VGABios' source code
5263                  <1>      ; vgabios-0.7a (2011)
5264                  <1>      ; by the LGPL VGABios developers Team (2001-2008)
5265                  <1>      ; 'vgabios.c', 'biosfn_load_text_8_16_pat'
5266                  <1>
5267                  <1>      ; biosfn_load_text_8_16_pat (AL,BL)
5268                  <1>
5269                  <1>      ; get_font_access();
5270                  <1>      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
5271                  <1>      ; for(i=0;i<0x100;i++)
5272                  <1>      ; {
5273                  <1>      ;     src = i * 16;
5274                  <1>      ;     dest = blockaddr + i * 32;
5275                  <1>      ;     memcpyb(0xA000, dest, 0xC000, vgafont16+src, 16);
5276                  <1>      ; }
5277                  <1>      ; release_font_access();
5278                  <1>      ; if(AL>=0x10)

```

```

5279          <1>      ; {
5280          <1>      ; set_scan_lines(16);
5281          <1>      ; }
5282          <1>
5283 00002D19 50          <1>      push    eax
5284 00002D1A E89EFEFFFF <1>      call   get_font_access
5285          <1>
5286          <1>      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
5287          <1>      ;mov    dl, bl
5288          <1>      ;and    dl, 3
5289          <1>      ;shl    dx, 14
5290          <1>      ;xchg   dx, bx
5291          <1>      ;and    dl, 4
5292          <1>      ;shl    dx, 11
5293          <1>      ;add    dx, bx
5294          <1>
5295          <1>      ;xor    dx, dx ; blockaddr = 0
5296          <1>      ; Always block 0 for TRDOS 386 ! (blockaddr=0(
5297          <1>
5298 00002D1F 28DB          <1>      sub     bl, bl ; i = 0
5299 00002D21 B710          <1>      mov     bh, 16
5300 00002D23 BE[A0220100] <1>      mov     esi, vgafont16
5301 00002D28 BF00000A00 <1>      mov     edi, 0A0000h
5302 00002D2D 0FB6C7      <1>      movzx   eax, bh
5303          <1>      lt8_16_1:
5304          <1>      ;mov    al, bl
5305          <1>      ;mul    bh
5306          <1>      ;movzx   esi, ax
5307          <1>      ;add    esi, vgafont16
5308          <1>      ;mov    al, bl ; i
5309          <1>      ;sub    ah, ah
5310          <1>      ;shl    ax, 5 ; * 32
5311          <1>      ;;add   ax, dx ; blockaddr + i * 32;
5312          <1>      ;movzx   edi, ax ; dest
5313          <1>      ;add    edi, 0A0000h
5314          <1>      ;movzx   ecx, bh
5315 00002D30 89C1          <1>      mov     ecx, eax ; 16
5316 00002D32 F3A4          <1>      rep     movsb
5317 00002D34 01C7          <1>      add     edi, eax ; add edi, 16
5318 00002D36 FEC3          <1>      inc     bl
5319 00002D38 75F6          <1>      jnz     short lt8_16_1
5320          <1>      ;
5321 00002D3A E8B3FEFFFF <1>      call   release_font_access
5322          <1>      ;
5323 00002D3F 58          <1>      pop     eax
5324          <1>      ; if(AL>=0x10)
5325 00002D40 3C10          <1>      cmp     al, 10h
5326 00002D42 7205          <1>      jnb     short lt8_16_2
5327          <1>      ; BH = 16
5328          <1>      ; set_scan_lines(16);
5329 00002D44 E8E8FEFFFF <1>      call   set_scan_lines
5330          <1>      lt8_16_2:
5331 00002D49 C3          <1>      retn
5332          <1>
5333          <1>      load_gfx_user_chars:
5334          <1>      ; 08/08/2016
5335          <1>      ; 10/07/2016
5336          <1>      ; Setup User-Defined Font for Graphics Mode (VGA)
5337          <1>      ;
5338          <1>      ; derived from 'Plex86/Bochs VGABios' source code
5339          <1>      ; vgabios-0.7a (2011)
5340          <1>      ; by the LGPL VGABios developers Team (2001-2008)
5341          <1>      ; 'vgabios.c', 'biosfn_load_gfx_user_chars'
5342          <1>
5343          <1>      ; biosfn_load_gfx_user_chars (ES,BP,CX,BL,DL)
5344          <1>      ; /* set 0x43 INT pointer */
5345          <1>      ; write_word(0x0, 0x43*4, BP);
5346          <1>      ; write_word(0x0, 0x43*4+2, ES);
5347 00002D4A 31C0          <1>      xor     eax, eax
5348 00002D4C 48          <1>      dec     eax ; 0FFFFFFFFh (user defined fonts)
5349 00002D4D A3[CE450100] <1>      mov     [VGA_INT43H], eax
5350          <1>
5351          <1>      ; BL    screen rows code: 00H = user-specified (in DL)
5352          <1>      ;                                01H = 14 rows
5353          <1>      ;                                02H = 25 rows
5354          <1>      ;                                03H = 43 rows
5355          <1>      ; CX    bytes per character definition
5356          <1>      ; DL    (when BL=0) custom number of character rows on screen
5357          <1>      ; dh = 0 -> 256 characters
5358          <1>      ; dh = 80h -> 128 characters
5359          <1>      ; (If DH <> 0 and DH <> 80h -> invalid)
5360          <1>      ; EBP    address of font-definition information (user's mem space)
5361          <1>
5362          <1>      ; switch (BL) {
5363          <1>      ; case 0:
5364          <1>      ;     write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
5365          <1>      ;     break;
5366 00002D52 20DB          <1>      and     bl, bl
5367 00002D54 7508          <1>      jnz     short l_gfx_uc_1
5368 00002D56 8815[2A5E0000] <1>      mov     [VGA_ROWS], dl ; not DL-1 !
5369 00002D5C EB23          <1>      jmp     short l_gfx_uc_4
5370          <1>      l_gfx_uc_1:
5371          <1>      ; case 1:
5372          <1>      ;     write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
5373          <1>      ;     break;
5374 00002D5E FECB          <1>      dec     bl
5375 00002D60 7509          <1>      jnz     short l_gfx_uc_2
5376          <1>      ; bl = 1
5377 00002D62 C605[2A5E0000]0E <1>      mov     byte [VGA_ROWS], 14 ; not 13 !
5378 00002D69 EB16          <1>      jmp     short l_gfx_uc_4
5379          <1>      l_gfx_uc_2:
5380 00002D6B FECB          <1>      dec     bl

```

```

5381 00002D6D 740B      <1>      jz      short l_gfx_uc_3 ; bl = 2
5382 00002D6F FECB      <1>      dec     bl
5383 00002D71 750E      <1>      jnz     short l_gfx_uc_4 ; bl > 3
5384                      <1>      ; bl = 3
5385                      <1>      ; case 3:
5386                      <1>      ;     write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
5387                      <1>      ;     break;
5388 00002D73 C605[2A5E0000]2B <1>      mov     byte [VGA_ROWS], 43 ; not 42 !
5389                      <1> l_gfx_uc_3:
5390                      <1>      ; case 2:
5391                      <1>      ; default:
5392                      <1>      ;     write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
5393                      <1>      ;     break;
5394                      <1>      ; bl = 2 or bl > 3
5395 00002D7A C605[2A5E0000]19 <1>      mov     byte [VGA_ROWS], 25 ; not 24 !
5396                      <1>      ; }
5397                      <1> l_gfx_uc_4:
5398                      <1>      ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, CX);
5399 00002D81 880D[265E0000] <1>      mov     [CHAR_HEIGHT], cl
5400                      <1>      ; }
5401 00002D87 C3          <1>      retn
5402                      <1>
5403                      <1> load_gfx_8_14_chars:
5404                      <1>      ; 08/08/2016
5405                      <1>      ; 10/07/2016
5406                      <1>      ; Setup ROM 8x14 Font for Graphics Mode (VGA)
5407                      <1>      ;
5408                      <1>      ; derived from 'Plex86/Bochs VGABios' source code
5409                      <1>      ; vgabios-0.7a (2011)
5410                      <1>      ; by the LGPL VGABios developers Team (2001-2008)
5411                      <1>      ; 'vgabios.c', 'biosfn_load_gfx_8_14_chars'
5412                      <1>
5413                      <1>      ; biosfn_load_gfx_8_14_chars (BL)
5414                      <1>      ; /* set 0x43 INT pointer */
5415                      <1>      ; write_word(0x0, 0x43*4, &vgafont14);
5416                      <1>      ; write_word(0x0, 0x43*4+2, 0xC000);
5417 00002D88 C705[CE450100]- <1>      mov     dword [VGA_INT43H], vgafont14
5418 00002D8E [A0140100]   <1>
5419                      <1>      ; BL      screen rows code: 00H = user-specified (in DL)
5420                      <1>      ;                               01H = 14 rows
5421                      <1>      ;                               02H = 25 rows
5422                      <1>      ;                               03H = 43 rows
5423                      <1>      ; DL      (when BL=0) custom number of char rows on screen
5424                      <1>
5425                      <1>      ; switch (BL) {
5426                      <1>      ; case 0:
5427                      <1>      ;     write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, DL-1);
5428                      <1>      ;     break;
5429 00002D92 20DB      <1>      and     bl, bl
5430 00002D94 7508      <1>      jnz     short l_gfx_8_14c_1
5431 00002D96 8815[2A5E0000] <1>      mov     [VGA_ROWS], dl ; not DL-1 !
5432 00002D9C EB23      <1>      jmp     short l_gfx_8_14c_4
5433                      <1> l_gfx_8_14c_1:
5434                      <1>      ; case 1:
5435                      <1>      ;     write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 13);
5436                      <1>      ;     break;
5437 00002D9E FECB      <1>      dec     bl
5438 00002DA0 7509      <1>      jnz     short l_gfx_8_14c_2
5439                      <1>      ; bl = 1
5440 00002DA2 C605[2A5E0000]0E <1>      mov     byte [VGA_ROWS], 14 ; not 13 !
5441 00002DA9 EB16      <1>      jmp     short l_gfx_8_14c_4
5442                      <1> l_gfx_8_14c_2:
5443 00002DAB FECB      <1>      dec     bl
5444 00002DAD 740B      <1>      jz      short l_gfx_8_14c_3 ; bl = 2
5445 00002DAF FECB      <1>      dec     bl
5446 00002DB1 750E      <1>      jnz     short l_gfx_8_14c_4 ; bl > 3
5447                      <1>      ; bl = 3
5448                      <1>      ; case 3:
5449                      <1>      ;     write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 42);
5450                      <1>      ;     break;
5451 00002DB3 C605[2A5E0000]2B <1>      mov     byte [VGA_ROWS], 43 ; not 42 !
5452                      <1> l_gfx_8_14c_3:
5453                      <1>      ; case 2:
5454                      <1>      ; default:
5455                      <1>      ;     write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, 24);
5456                      <1>      ;     break;
5457                      <1>      ; bl = 2 or bl > 3
5458 00002DBA C605[2A5E0000]19 <1>      mov     byte [VGA_ROWS], 25 ; not 24 !
5459                      <1>      ; }
5460                      <1> l_gfx_8_14c_4:
5461                      <1>      ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 14);
5462 00002DC1 C605[265E0000]0E <1>      mov     byte [CHAR_HEIGHT], 14
5463                      <1>      ; }
5464 00002DC8 C3          <1>      retn
5465                      <1>
5466                      <1> load_gfx_8_8_chars:
5467                      <1>      ; 08/08/2016
5468                      <1>      ; 10/07/2016
5469                      <1>      ; Setup ROM 8x14 Font for Graphics Mode (VGA)
5470                      <1>      ;
5471                      <1>      ; derived from 'Plex86/Bochs VGABios' source code
5472                      <1>      ; vgabios-0.7a (2011)
5473                      <1>      ; by the LGPL VGABios developers Team (2001-2008)
5474                      <1>      ; 'vgabios.c', 'biosfn_load_gfx_8_8_dd_chars'
5475                      <1>
5476                      <1>      ; biosfn_load_gfx_8_8_dd_chars (BL)
5477                      <1>      ; /* set 0x43 INT pointer */
5478                      <1>      ; write_word(0x0, 0x43*4, &vgafont8);
5479                      <1>      ; write_word(0x0, 0x43*4+2, 0xC000);
5480 00002DC9 C705[CE450100]- <1>      mov     dword [VGA_INT43H], vgafont8
5481 00002DCF [A00C0100]   <1>

```

```

5481 <1>
5482 <1> ; BL screen rows code: 00H = user-specified (in DL)
5483 <1> ; 01H = 14 rows
5484 <1> ; 02H = 25 rows
5485 <1> ; 03H = 43 rows
5486 <1> ; DL (when BL=0) custom number of char rows on screen
5487 <1>
5488 <1> ; switch (BL) {
5489 <1> ; case 0:
5490 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, DL-1);
5491 <1> ; break;
5492 00002DD3 20DB <1> and bl, bl
5493 00002DD5 7508 <1> jnz short l_gfx_8_8c_1
5494 00002DD7 8815[2A5E0000] <1> mov [VGA_ROWS], dl ; not DL-1 !
5495 00002DDD EB23 <1> jmp short l_gfx_8_8c_4
5496 <1> l_gfx_8_8c_1:
5497 <1> ; case 1:
5498 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 13);
5499 <1> ; break;
5500 00002DDF FECB <1> dec bl
5501 00002DE1 7509 <1> jnz short l_gfx_8_8c_2
5502 <1> ; bl = 1
5503 00002DE3 C605[2A5E0000]0E <1> mov byte [VGA_ROWS], 14 ; not 13 !
5504 00002DEA EB16 <1> jmp short l_gfx_8_8c_4
5505 <1> l_gfx_8_8c_2:
5506 00002DEC FECB <1> dec bl
5507 00002DEE 740B <1> jz short l_gfx_8_8c_3 ; bl = 2
5508 00002DF0 FECB <1> dec bl
5509 00002DF2 750E <1> jnz short l_gfx_8_8c_4 ; bl > 3
5510 <1> ; bl = 3
5511 <1> ; case 3:
5512 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 42);
5513 <1> ; break;
5514 00002DF4 C605[2A5E0000]2B <1> mov byte [VGA_ROWS], 43 ; not 42 !
5515 <1> l_gfx_8_8c_3:
5516 <1> ; case 2:
5517 <1> ; default:
5518 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 24);
5519 <1> ; break;
5520 <1> ; bl = 2 or bl > 3
5521 00002DFB C605[2A5E0000]19 <1> mov byte [VGA_ROWS], 25 ; not 24 !
5522 <1> ; }
5523 <1> l_gfx_8_8c_4:
5524 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 8);
5525 00002E02 C605[265E0000]08 <1> mov byte [CHAR_HEIGHT], 8
5526 <1> ; }
5527 00002E09 C3 <1> retn
5528 <1>
5529 <1> load_gfx_8_16_chars:
5530 <1> ; 08/08/2016
5531 <1> ; 10/07/2016
5532 <1> ; Setup ROM 8x14 Font for Graphics Mode (VGA)
5533 <1> ;
5534 <1> ; derived from 'Plex86/Bochs VGABios' source code
5535 <1> ; vgabios-0.7a (2011)
5536 <1> ; by the LGPL VGABios developers Team (2001-2008)
5537 <1> ; 'vgabios.c', 'biosfn_load_gfx_8_16_chars'
5538 <1>
5539 <1> ; biosfn_load_gfx_8_16_chars (BL)
5540 <1> ; /* set 0x43 INT pointer */
5541 <1> ; write_word(0x0, 0x43*4, &vgafont16);
5542 <1> ; write_word(0x0, 0x43*4+2, 0xC000);
5543 00002E0A C705[CE450100]- <1> mov dword [VGA_INT43H], vgafont16
5544 00002E10 [A0220100] <1>
5545 <1>
5546 <1> ; BL screen rows code: 00H = user-specified (in DL)
5547 <1> ; 01H = 14 rows
5548 <1> ; 02H = 25 rows
5549 <1> ; 03H = 43 rows
5550 <1> ; DL (when BL=0) custom number of char rows on screen
5551 <1>
5552 <1> ; switch (BL) {
5553 <1> ; case 0:
5554 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, DL-1);
5555 <1> ; break;
5556 00002E14 20DB <1> and bl, bl
5557 00002E16 7508 <1> jnz short l_gfx_8_16c_1
5558 00002E18 8815[2A5E0000] <1> mov [VGA_ROWS], dl ; not DL-1 !
5559 00002E1E EB23 <1> jmp short l_gfx_8_16c_4
5560 <1> l_gfx_8_16c_1:
5561 <1> ; case 1:
5562 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 13);
5563 <1> ; break;
5564 00002E20 FECB <1> dec bl
5565 00002E22 7509 <1> jnz short l_gfx_8_16c_2
5566 00002E24 C605[2A5E0000]0E <1> ; bl = 1
5567 00002E2B EB16 <1> mov byte [VGA_ROWS], 14 ; not 13 !
5568 <1> jmp short l_gfx_8_16c_4
5569 00002E2D FECB <1> l_gfx_8_16c_2:
5570 00002E2F 740B <1> dec bl
5571 00002E31 FECB <1> jz short l_gfx_8_16c_3 ; bl = 2
5572 00002E33 750E <1> dec bl
5573 <1> jnz short l_gfx_8_16c_4 ; bl > 3
5574 <1> ; bl = 3
5575 <1> ; case 3:
5576 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 42);
5577 00002E35 C605[2A5E0000]2B <1> ; break;
5578 <1> mov byte [VGA_ROWS], 43 ; not 42 !
5579 <1> l_gfx_8_16c_3:
5580 <1> ; case 2:
5581 <1> ; default:
5582 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 24);

```



```

5582      <1>      ;      break;
5583      <1>      ; bl = 2 or bl > 3
5584 00002E3C C605[2A5E0000]19      <1>      mov      byte [VGA_ROWS], 25      ; not 24 !
5585      <1>      ; }
5586      <1> l_gfx_8_16c_4:
5587      <1>      ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 16);
5588 00002E43 C605[265E0000]10      <1>      mov      byte [CHAR_HEIGHT], 16
5589      <1>      ; }
5590 00002E4A C3      <1>      retn
5591      <1>
5592      <1> get_font_info:
5593      <1>      ; 19/09/2016
5594      <1>      ; 08/08/2016
5595      <1>      ; 10/07/2016
5596      <1>      ; Get Current Character Generator Info (VGA)
5597      <1>      ;
5598      <1>      ; derived from 'Plex86/Bochs VGABios' source code
5599      <1>      ; vgabios-0.7a (2011)
5600      <1>      ; by the LGPL VGABios developers Team (2001-2008)
5601      <1>      ; 'vgabios.c', 'biosfn_get_font_info'
5602      <1>
5603      <1>      ; Modified for TRDOS 386 !
5604      <1>      ;
5605      <1>      ; INPUT ->
5606      <1>      ;      AX = 1130h
5607      <1>      ;      BL = 0 -> Get info for current VGA font
5608      <1>      ;      (BH = unused)
5609      <1>      ;      19/09/2016
5610      <1>      ;      BL > 0 -> Get requested character font data
5611      <1>      ;      BL = 1 -> vgafont8
5612      <1>      ;      BL = 2 -> vgafont14
5613      <1>      ;      BL = 3 -> vgafont16
5614      <1>      ;      BL > 3 -> Invalid function (for now!)
5615      <1>      ;      BH = ASCII code of the first character
5616      <1>      ;      ECX = Number of characters from the 1st char
5617      <1>      ;      ECX >= 256 -> All (256-BH) characters
5618      <1>      ;      ECX = 0 -> All characters (BH = unused)
5619      <1>      ;      EDX = User's Buffer Address
5620      <1>      ; OUTPUT ->
5621      <1>      ;      AL = height (scanlines), bytes per character
5622      <1>      ;      AH = screen rows
5623      <1>      ;      Byte 16-23 of EAX = number of columns
5624      <1>      ;      Byte 24-31 of EAX =
5625      <1>      ;      0 -> default font (not configured yet)
5626      <1>      ;      0FFh -> user defined font
5627      <1>      ;      14 = vgafont14
5628      <1>      ;      8 = vgafont8
5629      <1>      ;      16 = vgafont16
5630      <1>      ;      If BL input > 0 ->
5631      <1>      ;      EAX = Actual transfer count
5632      <1>      ;
5633 00002E4B 20DB      <1>      and      bl, bl
5634 00002E4D 7408      <1>      jz      short gfi_0
5635      <1>      ; invalid function (input)
5636 00002E4F 80FB03      <1>      cmp      bl, 3
5637 00002E52 7642      <1>      jna      short gfi_4
5638 00002E54 31C0      <1>      xor      eax, eax ; 0
5639 00002E56 C3      <1>      retn
5640      <1> gfi_0:
5641 00002E57 A0[265E0000]      <1>      mov      al, [CHAR_HEIGHT]
5642 00002E5C 8A25[2A5E0000]      <1>      mov      ah, [VGA_ROWS]
5643 00002E62 C1E010      <1>      shl      eax, 16
5644 00002E65 A0[245E0000]      <1>      mov      al, [CRT_COLS]
5645 00002E6A 8B0D[CE450100]      <1>      mov      ecx, [VGA_INT43H]
5646 00002E70 21C9      <1>      and      ecx, ecx
5647 00002E72 741E      <1>      jz      short gfi_2 ; 0 = default font
5648 00002E74 41      <1>      inc      ecx ; 0FFFFFFFFh -> 0 (user defined font)
5649 00002E75 7504      <1>      jnz      short gfi_1
5650 00002E77 FECC      <1>      dec      ah ; 0FFh
5651 00002E79 EB17      <1>      jmp      short gfi_2
5652      <1> gfi_1:
5653 00002E7B 49      <1>      dec      ecx ; 08/08/2016
5654 00002E7C B40E      <1>      mov      ah, 14
5655 00002E7E 81F9[A0140100]      <1>      cmp      ecx, vgafont14
5656 00002E84 740C      <1>      je      short gfi_2
5657 00002E86 B408      <1>      mov      ah, 8
5658 00002E88 81F9[A00C0100]      <1>      cmp      ecx, vgafont8
5659 00002E8E 7402      <1>      je      short gfi_2
5660      <1>      ; vgafont16
5661 00002E90 D0E4      <1>      shl      ah, 1 ; ah = 16
5662      <1> gfi_2:
5663 00002E92 C1C010      <1>      rol      eax, 16
5664      <1> gfi_3:
5665 00002E95 C3      <1>      retn
5666      <1> gfi_4:
5667 00002E96 89D7      <1>      mov      edi, edx ; **
5668 00002E98 80FB02      <1>      cmp      bl, 2
5669 00002E9B 720B      <1>      jb      short gfi_5
5670 00002E9D 772F      <1>      ja      short gfi_7
5671      <1>      ;BL = 2 -> vgafont14
5672 00002E9F BE[A0140100]      <1>      mov      esi, vgafont14 ; *
5673 00002EA4 B30E      <1>      mov      bl, 14
5674 00002EA6 EB07      <1>      jmp      short gfi_6
5675      <1> gfi_5:
5676      <1>      ;BL = 1 -> vgafont8
5677 00002EA8 BE[A00C0100]      <1>      mov      esi, vgafont8 ; *
5678 00002EAD B308      <1>      mov      bl, 8
5679      <1> gfi_6:
5680 0000EAF 09C9      <1>      or      ecx, ecx
5681 00002EB1 7424      <1>      jz      short gfi_8 ; all chars from the 00h
5682 00002EB3 88F8      <1>      mov      al, bh ; character index
5683 00002EB5 F6E3      <1>      mul      bl ; char index * char height/size

```

```

5684 00002EB7 0FB7D0      <1>      movzx  edx, ax
5685 00002EBA 01D6      <1>      add    esi, edx ; *
5686 00002EBC 66BAFF00    <1>      mov    dx, 255
5687 00002EC0 28FA      <1>      sub    dl, bh
5688 00002EC2 6642      <1>      inc    dx
5689 00002EC4 39D1      <1>      cmp    ecx, edx
5690 00002EC6 770F      <1>      ja     short gfi_8
5691 00002EC8 7412      <1>      je     short gfi_9
5692 00002ECA 89D1      <1>      mov    ecx, edx
5693 00002ECC EB0E      <1>      jmp    short gfi_9
5694
5695      <1> gfi_7:
5696 00002ECE BE[A0220100]    <1>      ;BL = 3 -> vgafontl6
5697 00002ED3 B310      <1>      mov    esi, vgafontl6 ; *
5698 00002ED5 EBD8      <1>      mov    bl, 16
5699      <1>      jmp    short gfi_6
5700 00002ED7 B900010000    <1> gfi_8:
5701      <1>      mov    ecx, 256
5702 00002EDC 6689C8    <1> gfi_9:
5703 00002EDF 30FF      <1>      mov    ax, cx ; character count
5704 00002EE1 66F7E3    <1>      xor    bh, bh
5705 00002EE4 6689C1    <1>      mul    bx ; char count * char height/size
5706      <1>      mov    cx, ax
5707      <1>      ; ESI = source address in system space
5708      <1>      ; EDI = user's buffer address
5709      <1>      ; ECX = transfer (byte) count
5710 00002EE7 E8E4B80000    <1>      call   transfer_to_user_buffer
5711 00002EEC 89C8      <1>      mov    eax, ecx ; actual transfer count
5712 00002EEE C3      <1>      retn
5713
5714      <1> vga_pal_funcs:
5715      <1>      ; 10/08/2016
5716      <1>      ; VGA Palette functions
5717      <1>      ;
5718      <1>      ; derived from 'Plex86/Bochs VGABios' source code
5719      <1>      ; vgabios-0.7a (2011)
5720      <1>      ; by the LGPL VGABios developers Team (2001-2008)
5721      <1>      ; 'vgabios.c', 'vgarom.asm'
5722
5723 00002EEF 3C00      <1>      cmp    al, 0
5724 00002EF1 0F848F000000    <1>      je     set_single_palette_reg
5725
5726 00002EF7 3C01      <1> vga_palf_1001:
5727 00002EF9 0F84B4000000    <1>      cmp    al, 1
5728      <1>      je     set_overscan_border_color
5729 00002EFF 3C02      <1> vga_palf_1002:
5730 00002F01 0F84B0000000    <1>      cmp    al, 2
5731      <1>      je     set_all_palette_reg
5732 00002F07 3C03      <1> vga_palf_1003:
5733 00002F09 0F84E8000000    <1>      cmp    al, 3
5734      <1>      je     toggle_intensity
5735 00002F0F 3C07      <1> vga_palf_1007:
5736 00002F11 0F840D010000    <1>      cmp    al, 7
5737 00002F17 7266      <1>      je     get_single_palette_reg
5738      <1>      jnb    short vga_palf_unknown
5739 00002F19 3C08      <1> vga_palf_1008:
5740 00002F1B 0F8437010000    <1>      cmp    al, 8
5741      <1>      je     read_overscan_border_color
5742 00002F21 3C09      <1> vga_palf_1009:
5743 00002F23 0F8433010000    <1>      cmp    al, 9
5744      <1>      je     get_all_palette_reg
5745 00002F29 3C10      <1> vga_palf_1010:
5746 00002F2B 0F8487010000    <1>      cmp    al, 10h
5747 00002F31 724C      <1>      je     set_single_dac_reg
5748      <1>      jnb    short vga_palf_unknown
5749 00002F33 3C12      <1> vga_palf_1012:
5750 00002F35 0F8498010000    <1>      cmp    al, 12h
5751 00002F3B 7242      <1>      je     set_all_dac_reg
5752      <1>      jnb    short vga_palf_unknown
5753 00002F3D 3C13      <1> vga_palf_1013:
5754 00002F3F 0F84CC010000    <1>      cmp    al, 13h
5755      <1>      je     select_video_dac_color_page
5756 00002F45 3C15      <1> vga_palf_1015:
5757 00002F47 0F8412020000    <1>      cmp    al, 15h
5758 00002F4D 7230      <1>      je     read_single_dac_reg
5759      <1>      jnb    short vga_palf_unknown
5760 00002F4F 3C17      <1> vga_palf_1017:
5761 00002F51 0F8428020000    <1>      cmp    al, 17h
5762 00002F57 7226      <1>      je     read_all_dac_reg
5763      <1>      jnb    short vga_palf_unknown
5764 00002F59 3C18      <1> vga_palf_1018:
5765 00002F5B 0F845E020000    <1>      cmp    al, 18h
5766      <1>      je     set_pel_mask
5767 00002F61 3C19      <1> vga_palf_1019:
5768 00002F63 0F8462020000    <1>      cmp    al, 19h
5769      <1>      je     read_pel_mask
5770 00002F69 3C1A      <1> vga_palf_101A:
5771 00002F6B 0F8468020000    <1>      cmp    al, 1Ah
5772      <1>      je     read_video_dac_state
5773 00002F71 3C1B      <1> vga_palf_101B:
5774      <1>      cmp    al, 1Bh
5775      <1>      ;jne    short vga_palf_unknown
5776      <1>      ja     short vga_palf_unknown
5777 00002F75 E80CF7FFFF    <1>      call   gray_scale_summing
5778 00002F7A E9D5E5FFFF    <1>      jmp    VIDEO_RETURN
5779
5780      <1> vga_palf_unknown:
5781 00002F7F 29C0      <1>      sub    eax, eax ; 0 = invalid function
5782 00002F81 E9D3E5FFFF    <1>      jmp    _video_return
5783
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      0000FAA 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 E852B80000 <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      0000FEA 66BADA03    <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
5859      0000FEE EC      <1>      in      al, dx
5860      <1>      ; endif ; VBOX
5861      0000FEF 83C414      <1>      add     esp, 20
5862      0000FF2 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      0000FF7 66BADA03    <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
5873      0000FFB EC      <1>      in      al, dx
5874      0000FFC 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>      ; EDI = 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      <1>      mov    dx, 3DAh ; VGAREG_ACTL_RESET
5949      <1>      in     al, dx
5950      <1>      mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5951      <1>      mov    al, cl
5952      <1>      out    dx, al
5953      <1>      mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
5954      <1>      in     al, dx
5955      <1>      mov    [ebx], al
5956      <1>      inc    ebx
5957      <1>      inc    cl
5958      <1>      cmp    cl, 10h
5959      <1>      jne     short get_palette_loop
5960      <1>      mov    dx, 3DAh ; VGAREG_ACTL_RESET
5961      <1>      in     al, dx
5962      <1>      mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5963      <1>      mov    al, 11h
5964      <1>      out    dx, al
5965      <1>      mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
5966      <1>      in     al, dx
5967      <1>      mov    [ebx], al
5968      <1>      mov    dx, 3DAh ; VGAREG_ACTL_RESET
5969      <1>      in     al, dx
5970      <1>      mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
5971      <1>      mov    al, 20h
5972      <1>      out    dx, al
5973      <1>      ; ifdef VBOX
5974      <1>      mov    dx, 3DAh ; VGAREG_ACTL_RESET
5975      <1>      in     al, dx
5976      <1>      ; endif ; VBOX
5977      <1>
5978      <1>      mov    ecx, 17 ; transfer (byte) count
5979      <1>      ; ESI = source address in system space
5980      <1>      ; EDI = user's buffer address
5981      <1>      call   transfer_to_user_buffer
5982      <1>
5983      <1>      add     esp, 20
5984      <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 E82DB70000   <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          <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          <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, 0FFFFCh ; (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 E818B60000 <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
6185                <1>
6186                <1> read_video_dac_state:
6187                <1>      ; 10/08/2016
6188                <1>      ; Query DAC Color Paging State
6189                <1>      ; Output:
6190                <1>      ; BH = current active DAC color page
6191                <1>      ; BL = current active DAC paging mode
6192                <1>
6193 000031D9 66BADA03 <1>      mov    dx, 3DAh ; VGAREG_ACTL_RESET

```

```

6194 000031DD EC          <1>      in      al, dx
6195 000031DE 66BAC003    <1>      mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
6196 000031E2 B010        <1>      mov     al, 10h
6197 000031E4 EE          <1>      out     dx, al
6198 000031E5 66BAC103    <1>      mov     dx, 3C1h ; VGAREG_ACTL_READ_DATA
6199 000031E9 EC          <1>      in      al, dx
6200 000031EA 88C3        <1>      mov     bl, al
6201 000031EC C0EB07      <1>      shr     bl, 7
6202 000031EF 66BADA03    <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
6203 000031F3 EC          <1>      in      al, dx
6204 000031F4 66BAC003    <1>      mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
6205 000031F8 B014        <1>      mov     al, 14h
6206 000031FA EE          <1>      out     dx, al
6207 000031FB 66BAC103    <1>      mov     dx, 3C1h ; VGAREG_ACTL_READ_DATA
6208 000031FF EC          <1>      in      al, dx
6209 00003200 88C7        <1>      mov     bh, al
6210 00003202 80E70F      <1>      and     bh, 0Fh
6211 00003205 F6C301      <1>      test    bl, 01
6212 00003208 7503        <1>      jnz     short get_dac_16_page
6213 0000320A C0EF02      <1>      shr     bh, 2
6214                                <1> get_dac_16_page:
6215 0000320D 66BADA03    <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
6216 00003211 EC          <1>      in      al, dx
6217 00003212 66BAC003    <1>      mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
6218 00003216 B020        <1>      mov     al, 20h
6219 00003218 EE          <1>      out     dx, al
6220                                <1>      ; ifdef VBOX
6221 00003219 66BADA03    <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
6222 0000321D EC          <1>      in      al, dx
6223                                <1>      ; endif ; VBOX
6224 0000321E 66895C240C    <1>      mov     [esp+12], bx ; bx
6225 00003223 E92CE3FFFF    <1>      jmp     VIDEO_RETURN
6226                                <1>
6227                                <1> ; % include 'vidata.s' ; VIDEO DATA
6228                                <1>
6229                                <1> ; /// End Of VIDEO FUNCTIONS ///
1941
1942
1943
1944 00003228 FA          setup_rtc_int:
1945                                ; source: http://wiki.osdev.org/RTC
1946                                cli          ; disable interrupts
1947                                ; default int frequency is 1024 Hz (Lower 4 bits of register A is 0110b or 6)
1948                                ; in order to change this ...
1949                                ; frequency = 32768 >> (rate-1) --> 32768 >> 5 = 1024
1950                                ; (rate must be above 2 and not over 15)
1951                                ; new rate = 15 --> 32768 >> (15-1) = 2 Hz
1952 00003229 B08A        mov     al, 8Ah
1953 0000322B E670        out     70h, al ; set index to register A, disable NMI
1954 0000322D 90          nop
1955 0000322E E471        in      al, 71h ; get initial value of register A
1956 00003230 88C4        mov     ah, al
1957 00003232 80E4F0      and     ah, 0F0h
1958 00003235 B08A        mov     al, 8Ah
1959 00003237 E670        out     70h, al ; reset index to register A
1960 00003239 88E0        mov     al, ah
1961 0000323B 0C0F        or      al, 0Fh ; new rate (0Fh -> 15)
1962 0000323D E671        out     71h, al ; write only our rate to A. Note, rate is the bottom 4 bits.
1963                                ; enable RTC interrupt
1964 0000323F B08B        mov     al, 8Bh ;
1965 00003241 E670        out     70h, al ; select register B and disable NMI
1966 00003243 90          nop
1967 00003244 E471        in      al, 71h ; read the current value of register B
1968 00003246 88C4        mov     ah, al ;
1969 00003248 B08B        mov     al, 8Bh ;
1970 0000324A E670        out     70h, al ; set the index again (a read will reset the index to register B)
1971 0000324C 88E0        mov     al, ah ;
1972 0000324E 0C40        or      al, 40h ;
1973 00003250 E671        out     71h, al ; write the previous value ORed with 0x40. This turns on bit 6 of register B
1974 00003252 FB          sti
1975 00003253 C3          retn
1976
1977                                ; Write memory information
1978                                ; 29/01/2016
1979                                ; 06/11/2014
1980                                ; 14/08/2015
1981                                memory_info:
1982 00003254 A1[24380100] mov     eax, [memory_size] ; in pages
1983 00003259 50          push    eax
1984 0000325A C1E00C      shl     eax, 12 ; in bytes
1985 0000325D BB0A000000 mov     ebx, 10
1986 00003262 89D9        mov     ecx, ebx ; 10
1987 00003264 BE[6D060100] mov     esi, mem_total_b_str
1988 00003269 E8BD000000 call    bintdstr
1989 0000326E 58          pop     eax
1990 0000326F B107        mov     cl, 7
1991 00003271 BE[91060100] mov     esi, mem_total_p_str
1992 00003276 E8B0000000 call    bintdstr
1993                                ; 14/08/2015
1994 0000327B E8C8000000 call    calc_free_mem
1995                                ; edx = calculated free pages
1996                                ; ecx = 0
1997 00003280 A1[28380100] mov     eax, [free_pages]
1998 00003285 39D0        cmp     eax, edx ; calculated free mem value
1999                                ; and initial free mem value are same or not?
2000 00003287 751D        jne     short pmim ; print mem info with '?' if not
2001 00003289 52          push    edx ; free memory in pages
2002                                ;mov     eax, edx
2003 0000328A C1E00C      shl     eax, 12 ; convert page count
2004                                ; to byte count
2005 0000328D B10A        mov     cl, 10
2006 0000328F BE[B1060100] mov     esi, free_mem_b_str
2007 00003294 E892000000 call    bintdstr
2008 00003299 58          pop     eax

```

```

2007 0000329A B107          mov     cl, 7
2008 0000329C BE[D5060100]   mov     esi, free_mem_p_str
2009 000032A1 E885000000     call    bintdstr
2010                          pmim:
2011 000032A6 BE[5B060100]   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[4F380100]   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[4F380100]   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                          ; dsctpm.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[38380100] 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
2158          1 <1> ; *****
2159          2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskio.s
2160          3 <1> ; -----
2161          4 <1> ; Last Update: 15/01/2017
2162          5 <1> ; -----
2163          6 <1> ; Beginning: 24/01/2016
2164          7 <1> ; -----
2165          8 <1> ; Assembler: NASM version 2.11 (trdos386.s)
2166          9 <1> ; -----
2167          10 <1> ; Turkish Rational DOS
2168          11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
2169          12 <1> ;
2170          13 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
2171          14 <1> ; diskio.inc (22/08/2015)
2172          15 <1> ;
2173          16 <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
2174          17 <1> ; *****
2175          18 <1> ;
2176          19 <1> ; Retro UNIX 386 v1 Kernel - DISKIO.INC
2177          20 <1> ; Last Modification: 22/08/2015
2178          21 <1> ; (Initialized Disk Parameters Data is in 'DISKDATA.INC')
2179          22 <1> ; (Uninitialized Disk Parameters Data is in 'DISKBSS.INC')
2180          23 <1> ;
2181          24 <1> ; DISK I/O SYSTEM - Erdogan Tan (Retro UNIX 386 v1 project)
2182          25 <1> ;
2183          26 <1> ; ////////// DISK I/O SYSTEM //////////
2184          27 <1> ;
2185          28 <1> ; 06/02/2015
2186          29 <1> diskette_io:
2187          30 <1> pushfd
2188          31 <1> push cs
2189          32 <1> call DISKETTE_IO_1
2190          33 <1> retn
2191          34 <1> ;
2192          35 <1> ;;;; DISKETTE I/O ;;;; 06/02/2015 ;;;
2193          36 <1> ;////////////////////////////////////
2194          37 <1> ;
2195          38 <1> ; DISKETTE I/O - Erdogan Tan (Retro UNIX 386 v1 project)
2196          39 <1> ; 20/02/2015
2197          40 <1> ; 06/02/2015 (unix386.s)
2198          41 <1> ; 16/12/2014 - 02/01/2015 (dsectrm2.s)
2199          42 <1> ;
2200          43 <1> ; Code (DELAY) modifications - AWARD BIOS 1999 (ADISK.EQU, COMMON.MAC)
2201          44 <1> ;
2202          45 <1> ; ADISK.EQU
2203          46 <1> ;
2204          47 <1> ;----- Wait control constants
2205          48 <1> ;
2206          49 <1> ;amount of time to wait while RESET is active.
2207          50 <1> ;
2208          51 <1> WAITCPU_RESET_ON EQU 21 ;Reset on must last at least 14us
2209          52 <1> ;at 250 KBS xfer rate.

```



```

53 <1> ;see INTEL MCS, 1985, pg. 5-456
54 <1>
55 <1> WAITCPU_FOR_STATUS EQU 100 ;allow 30 microseconds for
56 <1> ;status register to become valid
57 <1> ;before re-reading.
58 <1>
59 <1> ;After sending a byte to NEC, status register may remain
60 <1> ;incorrectly set for 24 us.
61 <1>
62 <1> WAITCPU_RQM_LOW EQU 24 ;number of loops to check for
63 <1> ;RQM low.
64 <1>
65 <1> ; COMMON.MAC
66 <1> ;
67 <1> ; Timing macros
68 <1> ;
69 <1>
70 <1> %macro SIODELAY 0 ; SHORT IODELAY
71 <1> jmp short $+2
72 <1> %endmacro
73 <1>
74 <1> %macro IODELAY 0 ; NORMAL IODELAY
75 <1> jmp short $+2
76 <1> jmp short $+2
77 <1> %endmacro
78 <1>
79 <1> %macro NEWIODELAY 0
80 <1> out 0ebh,al
81 <1> %endmacro
82 <1>
83 <1> ; (According to) AWARD BIOS 1999 - ATORGS.ASM (dw -> equ, db -> equ)
84 <1> ;;; WAIT_FOR_MEM
85 <1> ;WAIT_FDU_INT_LO equ 017798 ; 2.5 secs in 30 micro units.
86 <1> ;WAIT_FDU_INT_HI equ 1
87 <1> ;WAIT_FDU_INT_LH equ 83334 ; 27/02/2015 (2.5 seconds waiting)
88 <1> ;;; WAIT_FOR_PORT
89 <1> ;WAIT_FDU_SEND_LO equ 16667 ; .5 secons in 30 us units.
90 <1> ;WAIT_FDU_SEND_HI equ 0
91 <1> ;WAIT_FDU_SEND_LH equ 16667 ; 27/02/2015
92 <1> ;Time to wait while waiting for each byte of NEC results = .5
93 <1> ;seconds. .5 seconds = 500,000 micros. 500,000/30 = 16,667.
94 <1> ;WAIT_FDU_RESULTS_LO equ 16667 ; .5 seconds in 30 micro units.
95 <1> ;WAIT_FDU_RESULTS_HI equ 0
96 <1> ;WAIT_FDU_RESULTS_LH equ 16667 ; 27/02/2015
97 <1> ;;; WAIT_REFRESH
98 <1> ;amount of time to wait for head settle, per unit in parameter
99 <1> ;table = 1 ms.
100 <1> ;WAIT_FDU_HEAD_SETTLE equ 33 ; 1 ms in 30 micro units.
101 <1>
102 <1>
103 <1> ; ////////////////////////////////// DISKETTE I/O //////////////////////////////////
104 <1>
105 <1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - POSTEQU.INC)
106 <1>
107 <1> ;-----
108 <1> ; EQUATES USED BY POST AND BIOS :
109 <1> ;-----
110 <1>
111 <1> ;----- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----
112 <1> ;PORT_A EQU 060H ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
113 <1> ;PORT_B EQU 061H ; PORT B READ/WRITE DIAGNOSTIC REGISTER
114 <1> ;REFRESH_BIT EQU 00010000B ; REFRESH TEST BIT
115 <1>
116 <1> ;-----
117 <1> ; CMOS EQUATES FOR THIS SYSTEM :
118 <1> ;-----
119 <1> ;CMOS_PORT EQU 070H ; I/O ADDRESS OF CMOS ADDRESS PORT
120 <1> ;CMOS_DATA EQU 071H ; I/O ADDRESS OF CMOS DATA PORT
121 <1> ;NMI EQU 10000000B ; DISABLE NMI INTERRUPTS MASK -
122 <1> ; HIGH BIT OF CMOS LOCATION ADDRESS
123 <1>
124 <1> ;----- CMOS TABLE LOCATION ADDRESS'S ## -----
125 <1> ;CMOS_DISKETTE EQU 010H ; DISKETTE DRIVE TYPE BYTE ;
126 <1> ; EQU 011H ; - RESERVED ;C
127 <1> ;CMOS_DISK EQU 012H ; FIXED DISK TYPE BYTE ;H
128 <1> ; EQU 013H ; - RESERVED ;E
129 <1> ;CMOS_EQUIP EQU 014H ; EQUIPMENT WORD LOW BYTE ;C
130 <1>
131 <1> ;----- DISKETTE EQUATES -----
132 <1> ;INT_FLAG EQU 10000000B ; INTERRUPT OCCURRENCE FLAG
133 <1> ;DSK_CHG EQU 10000000B ; DISKETTE CHANGE FLAG MASK BIT
134 <1> ;DETERMINED EQU 00010000B ; SET STATE DETERMINED IN STATE BITS
135 <1> ;HOME EQU 00010000B ; TRACK 0 MASK
136 <1> ;SENSE_DRV_ST EQU 00000100B ; SENSE DRIVE STATUS COMMAND
137 <1> ;TRK_SLAP EQU 030H ; CRASH STOP (48 TPI DRIVES)
138 <1> ;QUIET_SEEK EQU 00AH ; SEEK TO TRACK 10
139 <1> ;MAX_DRV EQU 2 ; MAX NUMBER OF DRIVES
140 <1> ;HD12_SETTLE EQU 15 ; 1.2 M HEAD SETTLE TIME
141 <1> ;HD320_SETTLE EQU 20 ; 320 K HEAD SETTLE TIME
142 <1> ;MOTOR_WAIT EQU 37 ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
143 <1>
144 <1> ;----- DISKETTE ERRORS -----
145 <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)
350 <1> ;-----
351 <1> ;      (AH)= 18H  SET MEDIA TYPE FOR FORMAT
352 <1> ;      INPUT REGISTERS
353 <1> ;      (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM TRACKS
354 <1> ;      (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
355 <1> ;      BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
356 <1> ;      (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHACKED)
357 <1> ;      OUTPUT REGISTERS:
358 <1> ;      (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,

```

```
359 <1> ; UNCHANGED IF (AH) IS NON-ZERO
360 <1> ; (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
361 <1> ; - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
362 <1> ; - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED
363 <1> ; - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
364 <1> ; -----
365 <1> ; DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
366 <1> ; THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
367 <1> ; ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
368 <1> ; ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
369 <1> ; IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
370 <1> ; CHANGE ERROR CODE
371 <1> ; IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
372 <1> ; TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
373 <1> ; IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
374 <1> ;
375 <1> ; DATA VARIABLE -- @DISK_POINTER
376 <1> ; DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
377 <1> ; -----
378 <1> ; OUTPUT FOR ALL FUNCTIONS
379 <1> ; AH = STATUS OF OPERATION
380 <1> ; STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE_STATUS
381 <1> ; VARIABLE IN THE DATA SEGMENT OF THIS MODULE
382 <1> ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
383 <1> ; TYPE AH=(15)).
384 <1> ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
385 <1> ; FOR READ/WRITE/VERIFY
386 <1> ; DS,BX,DX,CX PRESERVED
387 <1> ; NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
388 <1> ; ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
389 <1> ; ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
390 <1> ; THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
391 <1> ; PROBLEM IS NOT DUE TO MOTOR START-UP.
392 <1> ; -----
393 <1> ;
394 <1> ; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
395 <1> ;
396 <1> ; -----
397 <1> ; | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
398 <1> ; |---|
399 <1> ; |---|
400 <1> ; |---|
401 <1> ; |---|
402 <1> ; |---|
403 <1> ; |---|
404 <1> ; |---|
405 <1> ; |---|
406 <1> ; |---|
407 <1> ; |---|
408 <1> ; |---|
409 <1> ; |---|
410 <1> ; |---|
411 <1> ; |---|
412 <1> ; |---|
413 <1> ; |---|
414 <1> ; |---|
415 <1> ; |---|
416 <1> ; |---|
417 <1> ; |---|
418 <1> ; |---|
419 <1> ; |---|
420 <1> ; |---|
421 <1> ; |---|
422 <1> ; |---|
423 <1> ; |---|
424 <1> ; |---|
425 <1> ; |---|
426 <1> ; |---|
427 <1> ; |---|
428 <1> ; |---|
429 <1> ; STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)
430 <1> ; -----
431 <1> ; PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)
432 <1> ; -----
433 <1> ;
434 <1> ; struc MD
435 00000000 <res 00000001> <1> .SPEC1 resb 1 ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
436 00000001 <res 00000001> <1> .SPEC2 resb 1 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
437 00000002 <res 00000001> <1> .OFF_TIM resb 1 ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
438 00000003 <res 00000001> <1> .BYT_SEC resb 1 ; 512 BYTES/SECTOR
439 00000004 <res 00000001> <1> .SEC_TRK resb 1 ; EOT (LAST SECTOR ON TRACK)
440 00000005 <res 00000001> <1> .GAP resb 1 ; GAP LENGTH
441 00000006 <res 00000001> <1> .DTL resb 1 ; DTL
442 00000007 <res 00000001> <1> .GAP3 resb 1 ; GAP LENGTH FOR FORMAT
443 00000008 <res 00000001> <1> .FIL_BYT resb 1 ; FILL BYTE FOR FORMAT
444 00000009 <res 00000001> <1> .HD_TIM resb 1 ; HEAD SETTLE TIME (MILLISECONDS)
445 0000000A <res 00000001> <1> .STR_TIM resb 1 ; MOTOR START TIME (1/8 SECONDS)
446 0000000B <res 00000001> <1> .MAX_TRK resb 1 ; MAX. TRACK NUMBER
447 0000000C <res 00000001> <1> .RATE resb 1 ; DATA TRANSFER RATE
448 <1> ; endstruc
449 <1> ;
450 <1> BIT7OFF EQU 7FH
451 <1> BIT7ON EQU 80H
452 <1> ;
453 <1> ;;int13h: ; 16/02/2015
454 <1> ;; 16/02/2015 - 21/02/2015
455 <1> int40h:
456 00003373 9C <1> pushfd
457 00003374 0E <1> push cs
458 00003375 E801000000 <1> call DISKETTE_IO_1
459 0000337A C3 <1> retn
460 <1> ;
```



```

461      <1> DISKETTE_IO_1:
462      <1>
463      0000337B FB      <1>          STI                ; INTERRUPTS BACK ON
464      0000337C 55      <1>          PUSH     eBP          ; USER REGISTER
465      0000337D 57      <1>          PUSH     eDI          ; USER REGISTER
466      0000337E 52      <1>          PUSH     eDX          ; HEAD #, DRIVE # OR USER REGISTER
467      0000337F 53      <1>          PUSH     eBX          ; BUFFER OFFSET PARAMETER OR REGISTER
468      00003380 51      <1>          PUSH     eCX          ; TRACK #-SECTOR # OR USER REGISTER
469      00003381 89E5    <1>          MOV      eBP,eSP          ; BP => PARAMETER LIST DEP. ON AH
470      <1>
471      <1>
472      <1>
473      <1>
474      <1>
475      <1>
476      <1>
477      <1>
478      <1>
479      <1>
480      <1>
481      <1>
482      <1>
483      00003383 06      <1>          push     es ; 06/02/2015
484      00003384 1E      <1>          PUSH     DS          ; BUFFER SEGMENT PARM OR USER REGISTER
485      00003385 56      <1>          PUSH     eSI          ; USER REGISTERS
486      <1>          ;CALL    DDS          ; SEGMENT OF BIOS DATA AREA TO DS
487      <1>          ;mov     cx, cs
488      <1>          ;mov     ds, cx
489      00003386 66B91000 <1>          mov      cx, KDATA
490      0000338A 8ED9    <1>          mov      ds, cx
491      0000338C 8EC1    <1>          mov      es, cx
492      <1>
493      <1>          ;CMP     AH,(FNC_TAE-FNC_TAB)/2      ; CHECK FOR > LARGEST FUNCTION
494      0000338E 80FC19 <1>          cmp      ah,(FNC_TAE-FNC_TAB)/4      ; 18/02/2015
495      00003391 7202    <1>          JB      short OK_FUNC      ; FUNCTION OK
496      00003393 B414    <1>          MOV      AH,14H          ; REPLACE WITH KNOWN INVALID FUNCTION
497      <1> OK_FUNC:
498      00003395 80FC01 <1>          CMP      AH,1          ; RESET OR STATUS ?
499      00003398 760C    <1>          JBE      short OK_DRV      ; IF RESET OR STATUS DRIVE ALWAYS OK
500      0000339A 80FC08 <1>          CMP      AH,8          ; READ DRIVE PARMS ?
501      0000339D 7407    <1>          JZ       short OK_DRV      ; IF SO DRIVE CHECKED LATER
502      0000339F 80FA01 <1>          CMP      DL,1          ; DRIVES 0 AND 1 OK
503      000033A2 7602    <1>          JBE      short OK_DRV      ; IF 0 OR 1 THEN JUMP
504      000033A4 B414    <1>          MOV      AH,14H          ; REPLACE WITH KNOWN INVALID FUNCTION
505      <1> OK_DRV:
506      000033A6 31C9    <1>          xor      ecx, ecx
507      <1>          ;mov     esi, ecx ; 08/02/2015
508      000033A8 89CF    <1>          mov      edi, ecx ; 08/02/2015
509      000033AA 88E1    <1>          MOV      CL,AH          ; CL = FUNCTION
510      <1>          ;XOR     CH,CH          ; CX = FUNCTION
511      <1>          ;SHL     CL, 1          ; FUNCTION TIMES 2
512      000033AC C0E102 <1>          SHL      CL, 2 ; 20/02/2015 ; FUNCTION TIMES 4 (for 32 bit offset)
513      000033AF BB[E7330000] <1>          MOV      eBX,FNC_TAB      ; LOAD START OF FUNCTION TABLE
514      000033B4 01CB    <1>          ADD      eBX,eCX          ; ADD OFFSET INTO TABLE => ROUTINE
515      000033B6 88F4    <1>          MOV      AH,DH          ; AX = HEAD #, # OF SECTORS OR DASD TYPE
516      000033B8 30F6    <1>          XOR      DH,DH          ; DX = DRIVE #
517      000033BA 6689C6 <1>          MOV      SI,AX          ; SI = HEAD #, # OF SECTORS OR DASD TYPE
518      000033BD 6689D7 <1>          MOV      DI,DX          ; DI = DRIVE #
519      <1>
520      <1>          ; 11/12/2014
521      000033C0 8815[A455C0000] <1>          mov      [cfd], dl          ; current floppy drive (for 'GET_PARM')
522      <1>
523      000033C6 8A25[A8380100] <1>          MOV      AH, [DSKETTE_STATUS] ; LOAD STATUS TO AH FOR STATUS FUNCTION
524      000033CC C605[A8380100]00 <1>          MOV      byte [DSKETTE_STATUS],0 ; INITIALIZE FOR ALL OTHERS
525      <1>
526      <1> ;
527      <1> ;
528      <1> ;
529      <1> ;
530      <1> ;
531      <1> ;
532      <1> ;
533      <1> ;
534      <1> ;
535      <1> ;
536      <1> ;
537      <1> ;
538      <1> ;
539      <1> ;
540      <1> ;
541      <1> ;
542      <1> ;
543      <1>
544      000033D3 FF13    <1>          CALL     dword [eBX]          ; (AH) = @DSKETTE_STATUS
545      000033D5 5E      <1>          POP      eSI          ; CALL THE REQUESTED FUNCTION
546      000033D6 1F      <1>          POP      DS
547      000033D7 07      <1>          pop      es          ; 06/02/2015
548      000033D8 59      <1>          POP      eCX
549      000033D9 5B      <1>          POP      eBX
550      000033DA 5A      <1>          POP      eDX
551      000033DB 5F      <1>          POP      eDI
552      000033DC 89E5    <1>          MOV      eBP, eSP
553      000033DE 50      <1>          PUSH     eAX
554      000033DF 9C      <1>          PUSHF
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>          IRET
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[A6380100]             <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[A5380100]100        <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[A9380100]           <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:
652 000034B9 6659                     <1>      POP     CX              ; CLEAR STACK
653                                     <1> DR_ERR:
654 000034BB 800D[A8380100]20          <1>      OR      byte [DSKETTE_STATUS],BAD_NEC ; SET ERROR CODE
655 000034C2 EBFA                     <1>      JMP     SHORT RESBAC      ; RETURN FROM RESET
656                                     <1>
657                                     <1> ;-----
658                                     <1> ; DISK_STATUS      (AH = 01H)
659                                     <1> ;          DISKETTE STATUS.
660                                     <1> ;

```

```

661 <1> ; ON ENTRY: AH : STATUS OF PREVIOUS OPERATION
662 <1> ;
663 <1> ; ON EXIT: AH, @DSKETTE_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.
664 <1> ;-----
665 <1> DSK_STATUS:
666 000034C4 8825[A8380100] <1> MOV [DSKETTE_STATUS],AH ; PUT BACK FOR SETUP END
667 000034CA E801090000 <1> CALL SETUP_END ; VARIOUS CLEANUPS
668 000034CF 6689F3 <1> MOV BX,SI ; GET SAVED AL TO BL
669 000034D2 88D8 <1> MOV AL,BL ; PUT BACK FOR RETURN
670 000034D4 C3 <1> RETn
671 <1>
672 <1> ;-----
673 <1> ; DISK_READ (AH = 02H)
674 <1> ; DISKETTE READ.
675 <1> ;
676 <1> ; ON ENTRY: DI : DRIVE #
677 <1> ; SI-HI : HEAD #
678 <1> ; SI-LOW : # OF SECTORS
679 <1> ; ES : BUFFER SEGMENT
680 <1> ; [BP] : SECTOR #
681 <1> ; [BP+1] : TRACK #
682 <1> ; [BP+2] : BUFFER OFFSET
683 <1> ;
684 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
685 <1> ;-----
686 <1>
687 <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
688 <1>
689 <1> DSK_READ:
690 000034D5 8025[A6380100]7F <1> AND byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
691 000034DC 66B846E6 <1> MOV AX,0E646H ; AX = NEC COMMAND, DMA COMMAND
692 000034E0 E83C040000 <1> CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
693 000034E5 C3 <1> RETn
694 <1>
695 <1> ;-----
696 <1> ; DISK_WRITE (AH = 03H)
697 <1> ; DISKETTE WRITE.
698 <1> ;
699 <1> ; ON ENTRY: DI : DRIVE #
700 <1> ; SI-HI : HEAD #
701 <1> ; SI-LOW : # OF SECTORS
702 <1> ; ES : BUFFER SEGMENT
703 <1> ; [BP] : SECTOR #
704 <1> ; [BP+1] : TRACK #
705 <1> ; [BP+2] : BUFFER OFFSET
706 <1> ;
707 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
708 <1> ;-----
709 <1>
710 <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
711 <1>
712 <1> DSK_WRITE:
713 000034E6 66B84AC5 <1> MOV AX,0C54AH ; AX = NEC COMMAND, DMA COMMAND
714 000034EA 800D[A6380100]80 <1> OR byte [MOTOR_STATUS],10000000B ; INDICATE WRITE OPERATION
715 000034F1 E82B040000 <1> CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
716 000034F6 C3 <1> RETn
717 <1>
718 <1> ;-----
719 <1> ; DISK_VERF (AH = 04H)
720 <1> ; DISKETTE VERIFY.
721 <1> ;
722 <1> ; ON ENTRY: DI : DRIVE #
723 <1> ; SI-HI : HEAD #
724 <1> ; SI-LOW : # OF SECTORS
725 <1> ; ES : BUFFER SEGMENT
726 <1> ; [BP] : SECTOR #
727 <1> ; [BP+1] : TRACK #
728 <1> ; [BP+2] : BUFFER OFFSET
729 <1> ;
730 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
731 <1> ;-----
732 <1> DSK_VERF:
733 000034F7 8025[A6380100]7F <1> AND byte [MOTOR_STATUS],01111111B ; INDICATE A READ OPERATION
734 000034FE 66B842E6 <1> MOV AX,0E642H ; AX = NEC COMMAND, DMA COMMAND
735 00003502 E81A040000 <1> CALL RD_WR_VF ; COMMON READ/WRITE/VERIFY
736 00003507 C3 <1> RETn
737 <1>
738 <1> ;-----
739 <1> ; DISK_FORMAT (AH = 05H)
740 <1> ; DISKETTE FORMAT.
741 <1> ;
742 <1> ; ON ENTRY: DI : DRIVE #
743 <1> ; SI-HI : HEAD #
744 <1> ; SI-LOW : # OF SECTORS
745 <1> ; ES : BUFFER SEGMENT
746 <1> ; [BP] : SECTOR #
747 <1> ; [BP+1] : TRACK #
748 <1> ; [BP+2] : BUFFER OFFSET
749 <1> ; @DISK_POINTER POINTS TO THE PARAMETER TABLE OF THIS DRIVE
750 <1> ;
751 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
752 <1> ;-----
753 <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[A6380100]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_LASRATE ; 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      <1>      CALL    FMTDMA_SET          ; SET UP THE DMA FOR FORMAT
765      <1>      JC      short FM_DON        ; RETURN WITH ERROR
766      <1>      MOV     AH,04DH              ; ESTABLISH THE FORMAT COMMAND
767      <1>      CALL    NEC_INIT            ; INITIALIZE THE NEC
768      <1>      JC      short FM_DON        ; ERROR - EXIT
769      <1>      MOV     eAX, FM_DON        ; LOAD ERROR ADDRESS
770      <1>      PUSH    eAX                 ; PUSH NEC_OUT ERROR RETURN
771      <1>      MOV     DL,3                 ; BYTES/SECTOR VALUE TO NEC
772      <1>      CALL    GET_PARM
773      <1>      CALL    NEC_OUTPUT
774      <1>      MOV     DL,4                 ; SECTORS/TRACK VALUE TO NEC
775      <1>      CALL    GET_PARM
776      <1>      CALL    NEC_OUTPUT
777      <1>      MOV     DL,7                 ; GAP LENGTH VALUE TO NEC
778      <1>      CALL    GET_PARM
779      <1>      CALL    NEC_OUTPUT
780      <1>      MOV     DL,8                 ; FILLER BYTE TO NEC
781      <1>      CALL    GET_PARM
782      <1>      CALL    NEC_OUTPUT
783      <1>      POP     eAX                 ; THROW AWAY ERROR
784      <1>      CALL    NEC_TERM            ; TERMINATE, RECEIVE STATUS, ETC,
785      <1> FM_DON:
786      <1>      CALL    XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
787      <1>      CALL    SETUP_END           ; VARIOUS CLEANUPS
788      <1>      MOV     BX,SI                ; GET SAVED AL TO BL
789      <1>      MOV     AL,BL                ; PUT BACK FOR RETURN
790      <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:
800      <1>      MOV     AX,SI                ; RESTORE AL
801      <1>      MOV     AH,BAD_CMD           ; SET BAD COMMAND ERROR
802      <1>      MOV     [DSKETTE_STATUS],AH ; STORE IN DATA AREA
803      <1>      STC                          ; SET CARRY INDICATING ERROR
804      <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      <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      <1>      sub     edx, edx
849      <1>      mov     ax, [fd0_type]
850      <1>      and     ax, ax
851      <1>      jz      NON_DRV
852      <1>      inc     dl
853      <1>      and     ah, ah
854      <1>      jz      short STO_DL
855      <1>      inc     dl
856      <1> STO_DL:
857      <1>      ;MOV     [BP+4],DL              ; STORE NUMBER OF DRIVES
858      <1>      mov     [ebp+8], edx ; 20/02/2015
859      <1>      CMP     DI,1                    ; CHECK FOR VALID DRIVE
860      <1>      JA      short NON_DRV1          ; DRIVE INVALID
861      <1>      ;MOV     BYTE [BP+5],1          ; MAXIMUM HEAD NUMBER = 1
862      <1>      mov     byte [ebp+9], 1 ; 20/02/2015
863      <1>      CALL    CMOS_TYPE              ; RETURN DRIVE TYPE IN AL
864      <1>      ;;20/02/2015

```

```

865 <1> ;JC short CHK_EST ; IF CMOS BAD CHECKSUM ESTABLISHED
866 <1> ;OR AL,AL ; TEST FOR NO DRIVE TYPE
867 000035CA 740F <1> JZ short CHK_EST ; JUMP IF SO
868 000035CC E81B020000 <1> CALL DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
869 000035D1 7208 <1> JC short CHK_EST ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
870 <1> ;MOV [BP+2],AL ; STORE VALID CMOS DRIVE TYPE
871 <1> ;mov [ebp+4], al ; 06/02/2015
872 000035D3 8A4B04 <1> MOV CL, [eBX+MD.SEC_TRK] ; GET SECTOR/TRACK
873 000035D6 8A6B0B <1> MOV CH, [eBX+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
874 000035D9 EB36 <1> JMP SHORT STO_CX ; CMOS GOOD, USE CMOS
875 <1> CHK_EST:
876 000035DB 8AA7[B5380100] <1> MOV AH, [DSK_STATE+eDI] ; LOAD STATE FOR THIS DRIVE
877 000035E1 F6C410 <1> TEST AH,MED_DET ; CHECK FOR ESTABLISHED STATE
878 000035E4 7457 <1> JZ short NON_DRV1 ; CMOS BAD/INVALID OR UNESTABLISHED
879 <1> USE_EST:
880 000035E6 80E4C0 <1> AND AH,RATE_MSK ; ISOLATE STATE
881 000035E9 80FC80 <1> CMP AH,RATE_250 ; RATE 250 ?
882 000035EC 7570 <1> JNE short USE_EST2 ; NO, GO CHECK OTHER RATE
883 <1>
884 <1> ;----- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
885 <1>
886 000035EE B001 <1> MOV AL,01 ; DRIVE TYPE 1 (360KB)
887 000035F0 E8F7010000 <1> CALL DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
888 000035F5 8A4B04 <1> MOV CL, [eBX+MD.SEC_TRK] ; GET SECTOR/TRACK
889 000035F8 8A6B0B <1> MOV CH, [eBX+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
890 000035FB F687[B5380100]01 <1> TEST byte [DSK_STATE+eDI],TRK_CAPA ; 80 TRACK ?
891 00003602 740D <1> JZ short STO_CX ; MUST BE 360KB DRIVE
892 <1>
893 <1> ;----- IT IS 1.44 MB DRIVE
894 <1>
895 <1> PARM144:
896 00003604 B004 <1> MOV AL,04 ; DRIVE TYPE 4 (1.44MB)
897 00003606 E8E1010000 <1> CALL DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
898 0000360B 8A4B04 <1> MOV CL, [eBX+MD.SEC_TRK] ; GET SECTOR/TRACK
899 0000360E 8A6B0B <1> MOV CH, [eBX+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
900 <1> STO_CX:
901 00003611 894D00 <1> MOV [eBP],eCX ; SAVE POINTER IN STACK FOR RETURN
902 <1> ES_DI:
903 <1> ;MOV [BP+6],BX ; ADDRESS OF MEDIA/DRIVE PARM TABLE
904 <1> ;mov [ebp+12], ebx ; 06/02/2015
905 <1> ;MOV AX,CS ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
906 <1> ;MOV ES,AX ; ES IS SEGMENT OF TABLE
907 <1> ;
908 <1> ; 28/05/2016
909 <1> ; 27/05/2016
910 <1> ; return floppy disk parameters table to user
911 <1> ; in user's buffer, which is pointed by EBX
912 <1> ;
913 00003614 57 <1> push edi
914 00003615 8B7D04 <1> mov edi, [ebp+4] ; ebx (input), user's buffer address
915 00003618 0FB6C0 <1> movzx eax, al
916 0000361B 894504 <1> mov [ebp+4], eax ; ebx ; drive type (for floppy drives)
917 <1> ; 01/06/2016 (INT 33h, disk type return for floppy disks, in BL)
918 0000361E A3[B0450100] <1> mov [user_buffer], eax ; 01/06/2016 (overwrite ebx return value)
919 <1> ; (INT 33h, Function 08h will replace user's buffer addr with disk type!)
920 <1> ;
921 00003623 89DE <1> mov esi, ebx ; floppy disk parameter table (16 bytes)
922 00003625 B910000000 <1> mov ecx, 16 ; 16 bytes
923 0000362A E8A1B10000 <1> call transfer_to_user_buffer ; trdosk6.s (16/05/2016)
924 0000362F 5F <1> pop edi
925 <1> DP_OUT:
926 00003630 E85C020000 <1> CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
927 00003635 6631C0 <1> XOR AX,AX ; CLEAR
928 00003638 F8 <1> CLC
929 00003639 C3 <1> RETn
930 <1>
931 <1> ;----- NO DRIYE PRESENT HANDLER
932 <1>
933 <1> NON_DRV:
934 <1> ;MOV BYTE [BP+4],0 ; CLEAR NUMBER OF DRIVES
935 0000363A 895508 <1> mov [ebp+8], edx ; 0 ; 20/02/2015
936 <1> NON_DRV1:
937 0000363D 6681FF8000 <1> CMP DI,80H ; CHECK FOR FIXED MEDIA TYPE REQUEST
938 00003642 720C <1> JB short NON_DRV2 ; CONTINUE IF NOT REQUEST FALL THROUGH
939 <1>
940 <1> ;----- FIXED DISK REQUEST FALL THROUGH ERROR
941 <1>
942 00003644 E848020000 <1> CALL XLAT_OLD ; ELSE TRANSLATE TO COMPATIBLE MODE
943 00003649 6689F0 <1> MOV AX,SI ; RESTORE AL
944 0000364C B401 <1> MOV AH,BAD_CMD ; SET BAD COMMAND ERROR
945 0000364E F9 <1> STC
946 0000364F C3 <1> RETn
947 <1>
948 <1> NON_DRV2:
949 <1> ;XOR AX,AX ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
950 00003650 31C0 <1> xor eax, eax
951 00003652 66894500 <1> MOV [eBP],AX ; TRACKS, SECTORS/TRACK = 0
952 <1> ;MOV [BP+5],AH ; HEAD = 0
953 00003656 886509 <1> mov [ebp+9], ah ; 06/02/2015
954 <1> ;MOV [BP+6],AX ; OFFSET TO DISK_BASE = 0
955 00003659 89450C <1> mov [ebp+12], eax
956 <1> ;MOV ES,AX ; ES IS SEGMENT OF TABLE
957 0000365C EBD2 <1> JMP SHORT DP_OUT
958 <1>
959 <1> ;----- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
960 <1>
961 <1> USE_EST2:
962 0000365E B002 <1> MOV AL,02 ; DRIVE TYPE 2 (1.2MB)
963 00003660 E887010000 <1> CALL DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
964 00003665 8A4B04 <1> MOV CL, [eBX+MD.SEC_TRK] ; GET SECTOR/TRACK
965 00003668 8A6B0B <1> MOV CH, [eBX+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
966 0000366B 80FC40 <1> CMP AH,RATE_300 ; RATE 300 ?

```



```

967 0000366E 74A1      <1>      JZ      short STO_CX          ; MUST BE 1.2MB DRIVE
968 00003670 EB92      <1>      JMP      SHORT PARM144        ; ELSE, IT IS 1.44MB DRIVE
969                      <1>
970                      <1> ;-----
971                      <1> ; DISK_TYPE (AH = 15H)
972                      <1> ;   THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
973                      <1> ;
974                      <1> ;   ON ENTRY: DI = DRIVE #
975                      <1> ;
976                      <1> ;   ON EXIT:  AH = DRIVE TYPE, CY=0
977                      <1> ;-----
978                      <1> DSK_TYPE:
979 00003672 E8E9010000 <1>      CALL     XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
980 00003677 8A87[B5380100] <1>      MOV      AL, [DSK_STATE+eDI] ; GET PRESENT STATE INFORMATION
981 0000367D 08C0      <1>      OR       AL,AL          ; CHECK FOR NO DRIVE
982 0000367F 7418      <1>      JZ       short NO_DRV        ;
983 00003681 B401      <1>      MOV      AH,NOCHGLN        ; NO CHANGE LINE FOR 40 TRACK DRIVE
984 00003683 A801      <1>      TEST     AL,TRK_CAPA        ; IS THIS DRIVE AN 80 TRACK DRIVE?
985 00003685 7402      <1>      JZ       short DT_BACK        ; IF NO JUMP
986 00003687 B402      <1>      MOV      AH,CHGLN          ; CHANGE LINE FOR 80 TRACK DRIVE
987                      <1> DT_BACK:
988 00003689 6650      <1>      PUSH     AX          ; SAVE RETURN VALUE
989 0000368B E801020000 <1>      CALL     XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
990 00003690 6658      <1>      POP      AX          ; RESTORE RETURN VALUE
991 00003692 F8        <1>      CLC          ; NO ERROR
992 00003693 6689F3     <1>      MOV      BX,SI          ; GET SAVED AL TO BL
993 00003696 88D8      <1>      MOV      AL,BL          ; PUT BACK FOR RETURN
994 00003698 C3        <1>      RETn
995                      <1> NO_DRV:
996 00003699 30E4      <1>      XOR      AH,AH          ; NO DRIVE PRESENT OR UNKNOWN
997 0000369B EBEC      <1>      JMP      SHORT DT_BACK
998                      <1>
999                      <1> ;-----
1000                     <1> ; DISK_CHANGE (AH = 16H)
1001                     <1> ;   THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.
1002                     <1> ;
1003                     <1> ;   ON ENTRY:  DI = DRIVE #
1004                     <1> ;
1005                     <1> ;   ON EXIT:   AH = @DSKETTE_STATUS
1006                     <1> ;               00 - DISK CHANGE LINE INACTIVE, CY = 0
1007                     <1> ;               06 - DISK CHANGE LINE ACTIVE, CY = 1
1008                     <1> ;-----
1009                     <1> DSK_CHANGE:
1010 0000369D E8BE010000 <1>      CALL     XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
1011 000036A2 8A87[B5380100] <1>      MOV      AL, [DSK_STATE+eDI] ; GET MEDIA STATE INFORMATION
1012 000036A8 08C0      <1>      OR       AL,AL          ; DRIVE PRESENT ?
1013 000036AA 7422      <1>      JZ       short DC_NON        ; JUMP IF NO DRIVE
1014 000036AC A801      <1>      TEST     AL,TRK_CAPA        ; 80 TRACK DRIVE ?
1015 000036AE 7407      <1>      JZ       short SETIT          ; IF SO , CHECK CHANGE LINE
1016                     <1> DC0:
1017 000036B0 E88D0A0000 <1>      CALL     READ_DSCHNG        ; GO CHECK STATE OF DISK CHANGE LINE
1018 000036B5 7407      <1>      JZ       short FINIS          ; CHANGE LINE NOT ACTIVE
1019                     <1>
1020 000036B7 C605[A8380100]06 <1> SETIT:      MOV      byte [DSKETTE_STATUS], MEDIA_CHANGE ; INDICATE MEDIA REMOVED
1021                     <1>
1022 000036BE E8CE010000 <1> FINIS:      CALL     XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
1023 000036C3 E808070000 <1>      CALL     SETUP_END          ; VARIOUS CLEANUPS
1024 000036C8 6689F3     <1>      MOV      BX,SI          ; GET SAVED AL TO BL
1025 000036CB 88D8      <1>      MOV      AL,BL          ; PUT BACK FOR RETURN
1026 000036CD C3        <1>      RETn
1027                     <1> DC_NON:
1028 000036CE 800D[A8380100]80 <1>      OR       byte [DSKETTE_STATUS], TIME_OUT ; SET TIMEOUT, NO DRIVE
1029 000036D5 EBE7      <1>      JMP      SHORT FINIS
1030                     <1>
1031                     <1> ;-----
1032                     <1> ; FORMAT_SET (AH = 17H)
1033                     <1> ;   THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF MEDIA TO BE USED
1034                     <1> ;   FOR THE FOLLOWING FORMAT OPERATION.
1035                     <1> ;
1036                     <1> ;   ON ENTRY:  SI LOW = DASD TYPE FOR FORMAT
1037                     <1> ;               DI      = DRIVE #
1038                     <1> ;
1039                     <1> ;   ON EXIT:   @DSKETTE_STATUS REFLECTS STATUS
1040                     <1> ;               AH = @DSKETTE_STATUS
1041                     <1> ;               CY = 1 IF ERROR
1042                     <1> ;-----
1043                     <1> FORMAT_SET:
1044 000036D7 E884010000 <1>      CALL     XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
1045 000036DC 6656      <1>      PUSH     SI          ; SAVE DASD TYPE
1046 000036DE 6689F0     <1>      MOV      AX,SI          ; AH = ? , AL , DASD TYPE
1047 000036E1 30E4      <1>      XOR      AH,AH          ; AH , 0 , AL , DASD TYPE
1048 000036E3 6689C6     <1>      MOV      SI,AX          ; SI = DASD TYPE
1049 000036E6 80A7[B5380100]0F <1> AND      byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
1050 000036ED 664E      <1>      DEC      SI          ; CHECK FOR 320/360K MEDIA & DRIVE
1051 000036EF 7509      <1>      JNZ     short NOT_320        ; BYPASS IF NOT
1052 000036F1 808F[B5380100]90 <1> OR       byte [DSK_STATE+eDI], MED_DET+RATE_250 ; SET TO 320/360
1053 000036F8 EB48      <1>      JMP      SHORT S0
1054                     <1>
1055                     <1> NOT_320:
1056 000036FA E8B6030000 <1>      CALL     MED_CHANGE        ; CHECK FOR TIME_OUT
1057 000036FF 803D[A8380100]80 <1> CMP      byte [DSKETTE_STATUS], TIME_OUT
1058 00003706 743A      <1>      JZ       short S0          ; IF TIME OUT TELL CALLER
1059                     <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[B5380100]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[B5380100]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[B5380100]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[B5380100]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[B5380100] <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[A8380100]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[B5380100]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[A8380100]80 <1>      CMP      byte [DSKETTE_STATUS], TIME_OUT ; IF TIME OUT TELL CALLER
1124 00003774 746B      <1>      JE       short SM_RTN
1125 00003776 C605[A8380100]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[D05B0000] <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[D15B0000] <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[A8380100]0C <1>      MOV      byte [DSKETTE_STATUS], MD_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[B5380100]0F <1>      AND      byte [DSK_STATE+eDI], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
1170 000037DB 0887[B5380100] <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      <1>      CALL    XLAT_OLD                ; TRANSLATE STATE TO COMPATIBLE MODE
1175      <1>      CALL    SETUP_END              ; VARIOUS CLEANUPS
1176      <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      <1>      PUSH    AX
1193      <1>      PUSH    eCX
1194      <1>      XOR     eBX,eBX                ; BX = INDEX TO DR_TYPE TABLE
1195      <1>      MOV     eCX,DR_CNT            ; CX = LOOP COUNT
1196      <1>      TYPE_CHK:
1197      <1>      MOV     AH,[DR_TYPE+eBX]      ; GET DRIVE TYPE
1198      <1>      CMP     AL,AH                ; DRIVE TYPE MATCH?
1199      <1>      JE      short DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
1200      <1>      ;ADD    BX,3                ; CHECK NEXT DRIVE TYPE
1201      <1>      add     ebx, 5 ; 16/02/2015 (32 bit address modification)
1202      <1>      LOOP    TYPE_CHK
1203      <1>      ;
1204      <1>      mov     ebx, MD_TBL6          ; 1.44MB fd parameter table
1205      <1>      ; Default for GET_PARM (11/12/2014)
1206      <1>      ;
1207      <1>      STC                          ; DRIVE TYPE NOT FOUND IN TABLE
1208      <1>      JMP     SHORT TYPE_RTN
1209      <1>      DR_TYPE_VALID:
1210      <1>      MOV     eBX,[DR_TYPE+eBX+1]  ; BX = MEDIA TABLE
1211      <1>      TYPE_RTN:
1212      <1>      POP     eCX
1213      <1>      POP     AX
1214      <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      <1>      PUSH    eAX                ; SAVE AX
1226      <1>      MOV     eAX, SPECBAC        ; LOAD ERROR ADDRESS
1227      <1>      PUSH    eAX                ; PUSH NEC_OUT ERROR RETURN
1228      <1>      MOV     AH,03H              ; SPECIFY COMMAND
1229      <1>      CALL    NEC_OUTPUT          ; OUTPUT THE COMMAND
1230      <1>      SUB     DL,DL                ; FIRST SPECIFY BYTE
1231      <1>      CALL    GET_PARM            ; GET PARAMETER TO AH
1232      <1>      CALL    NEC_OUTPUT          ; OUTPUT THE COMMAND
1233      <1>      MOV     DL,1                ; SECOND SPECIFY BYTE
1234      <1>      CALL    GET_PARM            ; GET PARAMETER TO AH
1235      <1>      CALL    NEC_OUTPUT          ; OUTPUT THE COMMAND
1236      <1>      POP     eAX                ; POP ERROR RETURN
1237      <1>      SPECBAC:
1238      <1>      POP     eAX                ; RESTORE ORIGINAL AX VALUE
1239      <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      <1>      PUSH    eAX                ; SAVE RATE DATA
1251      <1>      MOV     eAX, SPEC_ESBAC      ; LOAD ERROR ADDRESS
1252      <1>      PUSH    eAX                ; PUSH NEC_OUT ERROR RETURN
1253      <1>      MOV     AH,03H              ; SPECIFY COMMAND
1254      <1>      CALL    NEC_OUTPUT          ; OUTPUT THE COMMAND
1255      <1>      MOV     AH, [eBX+MD.SPEC1]  ; GET 1ST SPECIFY BYTE
1256      <1>      CALL    NEC_OUTPUT          ; OUTPUT THE COMMAND
1257      <1>      MOV     AH, [eBX+MD.SPEC2]  ; GET SECOND SPECIFY BYTE
1258      <1>      CALL    NEC_OUTPUT          ; OUTPUT THE COMMAND
1259      <1>      POP     eAX                ; POP ERROR RETURN
1260      <1>      SPEC_ESBAC:
1261      <1>      POP     eAX                ; RESTORE ORIGINAL AX VALUE
1262      <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      <1>      CMP     eDI,1                ; VALID DRIVE

```

```
1273 00003863 7725 <1> JA short XN_OUT ; IF INVALID BACK
1274 00003865 80BF[B5380100]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[B4380100] <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[B5380100]F8 <1> AND byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA)
1282 00003884 0887[B5380100] <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[B5380100]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[B4380100] <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[B4380100] <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[B5380100] <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[B4380100] <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[B5380100] <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 IHU
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[B5380100]F8 <1> AND byte [DSK_STATE+eDI], ~(DRV_DET+FMT_CAPA+TRK_CAPA) ; CLEAR DRIVE
1363 0000391A 0887[B5380100] <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[B5380100] <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[B5380100]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[B5380100]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 rww_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> rww_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[D05B0000] <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[D15B0000] <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[EE5B0000] <1> MOV eBX, MD_TBL1 ; POINT TO 40 TRACK 250 KBS
1439 000039A6 F687[B5380100]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[085C0000] <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_LASRATE ; 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
1478 000039F9 E84A030000
1479 000039FE 6658
1480 00003A00 7305
1481 00003A02 E928FFFFFF
1482
1483 00003A07 E8F4020000
1484 00003A0C E887030000
1485
1486 00003A11 6650
1487 00003A13 E879FEFFFF
1488 00003A18 6658
1489 00003A1A E8B1030000
1490 00003A1F C3
1491
1492
1493
1494
1495
1496 00003A20 F687[B5380100]10
1497 00003A27 7537
1498 00003A29 66B84000
1499 00003A2D F687[B5380100]04
1500 00003A34 740D
1501 00003A36 F687[B5380100]02
1502 00003A3D 7504
1503 00003A3F 66B88080
1504
1505 00003A43 80A7[B5380100]1F
1506 00003A4A 08A7[B5380100]
1507 00003A50 8025[B0380100]F3
1508 00003A57 C0C804
1509 00003A5A 0805[B0380100]
1510
1511 00003A60 C3
1512
1513
1514
1515
1516
1517 00003A61 F687[B5380100]10
1518 00003A68 7546
1519 00003A6A E82C040000
1520
1521
1522 00003A6F 7440
1523 00003A71 FEC8
1524
1525 00003A73 8AA7[B5380100]
1526 00003A79 80E40F
1527 00003A7C 08C0
1528 00003A7E 7505
1529 00003A80 80CC90
1530 00003A83 EB25
1531
1532 00003A85 FEC8
1533 00003A87 7505
1534
1535 00003A89 80CC10
1536 00003A8C EB1C
1537
1538 00003A8E FEC8
1539 00003A90 750F
1540 00003A92 F6C404
1541 00003A95 7410
1542 00003A97 F6C402
1543 00003A9A 740B
1544 00003A9C 80CC50
1545 00003A9F EB09
1546
1547 00003AA1 FEC8
1548 00003AA3 750C
1549 00003AA5 EBE2
1550
1551 00003AA7 80CC90
1552
1553
1554 00003AAA 88A7[B5380100]
1555
1556 00003AB0 C3
1557
1558 00003AB1 30E4
1559 00003AB3 EBF5
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570 00003AB5 E888060000
1571 00003ABA 7447
1572 00003ABC 80A7[B5380100]EF
1573
1574
1575
1576
1577
1578 00003AC3 6689F9

<1> CHK_RET:
<1> CALL RETRY ; CHECK FOR, SETUP RETRY
<1> POP AX ; RESTORE READ/WRITE/VERIFY PARAMETER
<1> JNC short RWV_END ; CY = 0 NO RETRY
<1> JMP DO_AGAIN ; CY = 1 MEANS RETRY
<1> RWV_END:
<1> CALL DSTATE ; ESTABLISH STATE IF SUCCESSFUL
<1> CALL NUM_TRANS ; AL = NUMBER TRANSFERRED
<1> RWV_BAC: ; BAD DMA ERROR ENTRY
<1> PUSH AX ; SAVE NUMBER TRANSFERRED
<1> CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
<1> POP AX ; RESTORE NUMBER TRANSFERRED
<1> CALL SETUP_END ; VARIOUS CLEANUPS
<1> RETn
<1>
<1> ;-----
<1> ; SETUP_STATE: INITIALIZES START AND END RATES.
<1> ;-----
<1> SETUP_STATE:
<1> TEST byte [DSK_STATE+eDI], MED_DET ; MEDIA DETERMINED ?
<1> JNZ short J1C ; NO STATES IF DETERMINED
<1> MOV AX,(RATE_500*256)+RATE_300 ; AH = START RATE, AL = END RATE
<1> TEST byte [DSK_STATE+eDI],DRV_DET ; DRIVE ?
<1> JZ short AX_SET ; DO NOT KNOW DRIVE
<1> TEST byte [DSK_STATE+eDI], FMT_CAPA ; MULTI-RATE?
<1> JNZ short AX_SET ; JUMP IF YES
<1> MOV AX,RATE_250*257 ; START A END RATE 250 FOR 360 DRIVE
<1> AX_SET:
<1> AND byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP) ; TURN OFF THE RATE
<1> OR [DSK_STATE+eDI], AH ; RATE FIRST TO TRY
<1> AND byte [LASTRATE], ~STRT_MSK ; ERASE LAST TO TRY RATE BITS
<1> ROR AL,4 ; TO OPERATION LAST RATE LOCATION
<1> OR [LASTRATE], AL ; LAST RATE
<1> J1C:
<1> RETn
<1>
<1> ;-----
<1> ; FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
<1> ;-----
<1> FMT_INIT:
<1> TEST byte [DSK_STATE+eDI], MED_DET ; IS MEDIA ESTABLISHED
<1> JNZ short F1_OUT ; IF SO RETURN
<1> CALL CMOS_TYPE ; RETURN DRIVE TYPE IN AL
<1> ; ; 20/02/2015
<1> ; ; JC short CL_Drv ; ERROR IN CMOS ASSUME NO DRIVE
<1> jz short CL_Drv ; ; 20/02/2015
<1> DEC AL ; MAKE ZERO ORIGIN
<1> ; ; JS short CL_Drv ; NO DRIVE IF AL 0
<1> MOV AH, [DSK_STATE+eDI] ; AH = CURRENT STATE
<1> AND AH, ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR
<1> OR AL,AL ; CHECK FOR 360
<1> JNZ short N_360 ; IF 360 WILL BE 0
<1> OR AH,MED_DET+RATE_250 ; ESTABLISH MEDIA
<1> JMP SHORT SKP_STATE ; SKIP OTHER STATE PROCESSING
<1> N_360:
<1> DEC AL ; 1.2 M DRIVE
<1> JNZ short N_12 ; JUMP IF NOT
<1> F1_RATE:
<1> OR AH,MED_DET+RATE_500 ; SET FORMAT RATE
<1> JMP SHORT SKP_STATE ; SKIP OTHER STATE PROCESSING
<1> N_12:
<1> DEC AL ; CHECK FOR TYPE 3
<1> JNZ short N_720 ; JUMP IF NOT
<1> TEST AH,DRV_DET ; IS DRIVE DETERMINED
<1> JZ short ISNT_12 ; TREAT AS NON 1.2 DRIVE
<1> TEST AH,FMT_CAPA ; IS 1.2M
<1> JZ short ISNT_12 ; JUMP IF NOT
<1> OR AH,MED_DET+RATE_300 ; RATE 300
<1> JMP SHORT SKP_STATE ; CONTINUE
<1> N_720:
<1> DEC AL ; CHECK FOR TYPE 4
<1> JNZ short CL_Drv ; NO DRIVE, CMOS BAD
<1> JMP SHORT F1_RATE
<1> ISNT_12:
<1> OR AH,MED_DET+RATE_250 ; MUST BE RATE 250
<1>
<1> SKP_STATE:
<1> MOV [DSK_STATE+eDI], AH ; STORE AWAY
<1> F1_OUT:
<1> RETn
<1> CL_Drv:
<1> XOR AH,AH ; CLEAR STATE
<1> JMP SHORT SKP_STATE ; SAVE IT
<1>
<1> ;-----
<1> ; MED_CHANGE
<1> ; CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
<1> ; CHECKS MEDIA CHANGE AGAIN.
<1> ;
<1> ; ON EXIT: CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
<1> ; @DSKETTE_STATUS = ERROR CODE
<1> ;-----
<1> MED_CHANGE:
<1> CALL READ_DSKCHNG ; READ DISK CHANCE LINE STATE
<1> JZ short MC_OUT ; BYPASS HANDLING DISK CHANGE LINE
<1> AND byte [DSK_STATE+eDI], ~MED_DET ; CLEAR STATE FOR THIS DRIVE
<1>
<1> ;
<1> ; THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
<1> ; ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
<1> ; BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).
<1> ;
<1>
<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[A6380100] <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[A8380100]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[A8380100]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[B0380100]3F <1>      AND     byte [LASTRATE], ~SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
1617 00003B0E 8A87[B5380100] <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[B0380100] <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[B0380100] <1>      AND     AH, [LASTRATE]          ; GET LAST DATA RATE SELECTED
1639 00003B2F 8A87[B5380100] <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 Address = 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>
1672                                     <1> ;; 20/02/2015
1673 00003B3E 8B5504      <1>      mov     edx, [ebp+4]          ; Buffer address
1674 00003B41 F7C2000000FF <1>      test    edx, 0FF000000h          ; 16 MB limit (22/08/2015, bugfix)
1675 00003B47 756E      <1>      jnz     short dma_bnd_err_stc
1676                                     <1> ;
1677 00003B49 6650      <1>      push    ax             ; DMA command
1678 00003B4B 52        <1>      push    edx             ; *
1679 00003B4C B203      <1>      mov     dl, 3             ; GET BYTES/SECTOR PARAMETER
1680 00003B4E E851030000 <1>      call    GET_PARM          ;

```

```

1681 00003B53 88E1      <1>      mov     cl, ah                ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
1682 00003B55 6689F0    <1>      mov     ax, si                ; Sector count
1683 00003B58 88C4      <1>      mov     ah, al                ; AH = # OF SECTORS
1684 00003B5A 28C0      <1>      sub     al, al                ; AL = 0, AX = # SECTORS * 256
1685 00003B5C 66D1E8    <1>      shr     ax, 1                ; AX = # SECTORS * 128
1686 00003B5F 66D3E0    <1>      shl     ax, cl                ; SHIFT BY PARAMETER VALUE
1687 00003B62 6648      <1>      dec     ax                ; -1 FOR DMA VALUE
1688 00003B64 6689C1    <1>      mov     cx, ax
1689 00003B67 5A          <1>      pop     edx                ; *
1690 00003B68 6658      <1>      pop     ax
1691 00003B6A 3C42      <1>      cmp     al, 42h
1692 00003B6C 7507      <1>      jne     short NOT_VERF
1693 00003B6E BA0000FF00  <1>      mov     edx, 0FF0000h
1694 00003B73 EB08      <1>      jmp     short J33
1695                                <1> NOT_VERF:
1696 00003B75 6601CA    <1>      add     dx, cx                ; check for overflow
1697 00003B78 723E      <1>      jc      short dma_bnd_err
1698                                <1>      ;
1699 00003B7A 6629CA    <1>      sub     dx, cx                ; Restore start address
1700                                <1> J33:
1701 00003B7D FA          <1>      CLI                        ; DISABLE INTERRUPTS DURING DMA SET-UP
1702 00003B7E E60C      <1>      OUT     DMA+12,AL            ; SET THE FIRST/LA5T F/F
1703                                <1>      IODELAY                    ; WAIT FOR I/O
1703 00003B80 EB00      <2>      jmp     short $+2
1703 00003B82 EB00      <2>      jmp     short $+2
1704 00003B84 E60B      <1>      OUT     DMA+11,AL            ; OUTPUT THE MODE BYTE
1705 00003B86 89D0      <1>      mov     eax, edx            ; Buffer address
1706 00003B88 E604      <1>      OUT     DMA+4,AL            ; OUTPUT LOW ADDRESS
1707                                <1>      IODELAY                    ; WAIT FOR I/O
1707 00003B8A EB00      <2>      jmp     short $+2
1707 00003B8C EB00      <2>      jmp     short $+2
1708 00003B8E 88E0      <1>      MOV     AL,AH
1709 00003B90 E604      <1>      OUT     DMA+4,AL            ; OUTPUT HIGH ADDRESS
1710 00003B92 C1E810    <1>      shr     eax, 16
1711                                <1>      IODELAY                    ; I/O WAIT STATE
1711 00003B95 EB00      <2>      jmp     short $+2
1711 00003B97 EB00      <2>      jmp     short $+2
1712 00003B99 E681      <1>      OUT     081H,AL            ; OUTPUT highest BITS TO PAGE REGISTER
1713                                <1>      IODELAY
1713 00003B9B EB00      <2>      jmp     short $+2
1713 00003B9D EB00      <2>      jmp     short $+2
1714 00003B9F 6689C8    <1>      mov     ax, cx                ; Byte count - 1
1715 00003BA2 E605      <1>      OUT     DMA+5,AL            ; LOW BYTE OF COUNT
1716                                <1>      IODELAY                    ; WAIT FOR I/O
1716 00003BA4 EB00      <2>      jmp     short $+2
1716 00003BA6 EB00      <2>      jmp     short $+2
1717 00003BA8 88E0      <1>      MOV     AL, AH
1718 00003BAA E605      <1>      OUT     DMA+5,AL            ; HIGH BYTE OF COUNT
1719                                <1>      IODELAY
1719 00003BAC EB00      <2>      jmp     short $+2
1719 00003BAE EB00      <2>      jmp     short $+2
1720 00003BB0 FB          <1>      STI                        ; RE-ENABLE INTERRUPTS
1721 00003BB1 B002      <1>      MOV     AL, 2                ; MODE FOR 8237
1722 00003BB3 E60A      <1>      OUT     DMA+10, AL          ; INITIALIZE THE DISKETTE CHANNEL
1723                                <1>
1724 00003BB5 F8          <1>      cld      ; 04/02/2016
1725 00003BB6 C3          <1>      retn
1726                                <1>
1727                                <1> dma_bnd_err_stc:
1728 00003BB7 F9          <1>      stc
1729                                <1> dma_bnd_err:
1730 00003BB8 C605[A8380100]09 <1>      MOV     byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
1731 00003BBF C3          <1>      RETn                        ; CY SET BY ABOVE IF ERROR
1732                                <1>
1733                                <1> ;; 16/12/2014
1734                                <1> ;; CLI                        ; DISABLE INTERRUPTS DURING DMA SET-UP
1735                                <1> ;; OUT     DMA+12,AL            ; SET THE FIRST/LA5T F/F
1736                                <1> ;; ;JMP     $+2                ; WAIT FOR I/O
1737                                <1> ;; IODELAY
1738                                <1> ;; OUT     DMA+11,AL            ; OUTPUT THE MODE BYTE
1739                                <1> ;; ;SIODELAY
1740                                <1> ;; ;CMP AL, 42H                ; DMA VERIFY COMMAND
1741                                <1> ;; ;JNE short NOT_VERF            ; NO
1742                                <1> ;; ;XOR AX, AX                ; START ADDRESS
1743                                <1> ;; ;JMP SHORT J33
1744                                <1> ;; ;NOT_VERF:
1745                                <1> ;; ;MOV     AX,ES                ; GET THE ES VALUE
1746                                <1> ;; ;ROL     AX,4                ; ROTATE LEFT
1747                                <1> ;; ;MOV     CH,AL                ; GET HIGHEST NIBBLE OF ES TO CH
1748                                <1> ;; ;AND     AL,11110000B            ; ZERO THE LOW NIBBLE FROM SEGMENT
1749                                <1> ;; ;ADD     AX,[BP+2]            ; TEST FOR CARRY FROM ADDITION
1750                                <1> ;; mov     eax, [ebp+4] ; 06/02/2015
1751                                <1> ;; ;JNC     short J33
1752                                <1> ;; ;INC     CH                ; CARRY MEANS HIGH 4 BITS MUST BE INC
1753                                <1> ;; ;J33:
1754                                <1> ;; PUSH     eAX                ; SAVE START ADDRESS
1755                                <1> ;; OUT     DMA+4,AL            ; OUTPUT LOW ADDRESS
1756                                <1> ;; ;JMP     $+2                ; WAIT FOR I/O
1757                                <1> ;; IODELAY
1758                                <1> ;; MOV     AL,AH
1759                                <1> ;; OUT     DMA+4,AL            ; OUTPUT HIGH ADDRESS
1760                                <1> ;; shr     eax, 16                ; 07/02/2015
1761                                <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, 0FFF0000h ; 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, 0FFF0000h ; 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 fmdtma_bnd_err ; 08/02/2015
1914 <1> ;; and ecx, 0FFF0000h ; 16 MB limit
1915 <1> ;; jz short FMTDMA_OK
1916 <1> ;; stc ; 20/02/2015
1917 <1> ;; fmdtma_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 FROM 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[A9380100] <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[A8380100] <1> OR [DSKETTE_STATUS], AH
2046 <1> SET_END:
2047 00003CF3 803D[A8380100]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>
2056 <1> ;-----
2057 <1> ; DSTATE: ESTABLISH STATE UPON SUCCESSFUL OPERATION.
2058 <1> ;-----
2059 <1> DSTATE:
2060 00003D00 803D[A8380100]00 <1> CMP byte [DSKETTE_STATUS],0 ; CHECK FOR ERROR
2061 00003D07 753E <1> JNZ short SETBAC ; IF ERROR JUMP
2062 00003D09 808F[B5380100]10 <1> OR byte [DSK_STATE+eDI],MED_DET ; NO ERROR, MARK MEDIA AS DETERMINED
2063 00003D10 F687[B5380100]04 <1> TEST byte [DSK_STATE+eDI],DRV_DET ; DRIVE DETERMINED ?
2064 00003D17 752E <1> JNZ short SETBAC ; IF DETERMINED NO TRY TO DETERMINE

```

```

2065 00003D19 8A87[B5380100] <1>      MOV     AL,[DSK_STATE+eDI] ; LOAD STATE
2066 00003D1F 24C0 <1>      AND     AL,RATE_MSK ; KEEP ONLY RATE
2067 00003D21 3C80 <1>      CMP     AL,RATE_250 ; RATE 250 ?
2068 00003D23 751B <1>      JNE     short M_12 ; NO, MUST BE 1.2M OR 1.44M DRIVE
2069 <1>
2070 <1> ;----- CHECK IF IT IS 1.44M
2071 <1>
2072 00003D25 E871010000 <1>      CALL    CMOS_TYPE ; RETURN DRIVE TYPE IN (AL)
2073 <1>      ;;20/02/2015
2074 <1>      ;;JC     short M_12 ; CMOS BAD
2075 00003D2A 7414 <1>      jz      short M_12 ;; 20/02/2015
2076 00003D2C 3C04 <1>      CMP     AL, 4 ; 1.44MB DRIVE ?
2077 00003D2E 7410 <1>      JE      short M_12 ; YES
2078 <1> M_720:
2079 00003D30 80A7[B5380100]FD <1>      AND     byte [DSK_STATE+eDI], ~FMT_CAPA ; TURN OFF FORMAT CAPABILITY
2080 00003D37 808F[B5380100]04 <1>      OR      byte [DSK_STATE+eDI],DRV_DET ; MARK DRIVE DETERMINED
2081 00003D3E EB07 <1>      JMP     SHORT SETBAC ; BACK
2082 <1> M_12:
2083 00003D40 808F[B5380100]06 <1>      OR      byte [DSK_STATE+eDI],DRV_DET+FMT_CAPA
2084 <1> ; TURN ON DETERMINED & FMT CAPA
2085 <1> SETBAC:
2086 00003D47 C3 <1>      RETn
2087 <1>
2088 <1> ;-----
2089 <1> ; RETRY
2090 <1> ; DETERMINES WHETHER A RETRY IS NECESSARY.
2091 <1> ; IF RETRY IS REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
2092 <1> ;
2093 <1> ; ON EXIT: CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
2094 <1> ;-----
2095 <1> RETRY:
2096 00003D48 803D[A8380100]00 <1>      CMP     byte [DSKETTE_STATUS],0 ; GET STATUS OF OPERATION
2097 00003D4F 7445 <1>      JZ      short NO_RETRY ; SUCCESSFUL OPERATION
2098 00003D51 803D[A8380100]80 <1>      CMP     byte [DSKETTE_STATUS],TIME_OUT ; IF TIME OUT NO RETRY
2099 00003D58 743C <1>      JZ      short NO_RETRY
2100 00003D5A 8AA7[B5380100] <1>      MOV     AH,[DSK_STATE+eDI] ; GET MEDIA STATE OF DRIVE
2101 00003D60 F6C410 <1>      TEST    AH,MED_DET ; ESTABLISHED/DETERMINED ?
2102 00003D63 7531 <1>      JNZ     short NO_RETRY ; IF ESTABLISHED STATE THEN TRUE ERROR
2103 00003D65 80E4C0 <1>      AND     AH,RATE_MSK ; ISOLATE RATE
2104 00003D68 8A2D[B0380100] <1>      MOV     CH,[LASTRATE] ; GET START OPERATION STATE
2105 00003D6E C0C504 <1>      ROL     CH,4 ; TO CORRESPONDING BITS
2106 00003D71 80E5C0 <1>      AND     CH,RATE_MSK ; ISOLATE RATE BITS
2107 00003D74 38E5 <1>      CMP     CH,AH ; ALL RATES TRIED
2108 00003D76 741E <1>      JE      short NO_RETRY ; IF YES, THEN TRUE ERROR
2109 <1>
2110 <1> ; SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
2111 <1> ; 00000000B (500) -> 10000000B (250)
2112 <1> ; 10000000B (250) -> 01000000B (300)
2113 <1> ; 01000000B (300) -> 00000000B (500)
2114 <1>
2115 00003D78 80FC01 <1>      CMP     AH,RATE_500+1 ; SET CY FOR RATE 500
2116 00003D7B D0DC <1>      RCR     AH,1 ; TO NEXT STATE
2117 00003D7D 80E4C0 <1>      AND     AH,RATE_MSK ; KEEP ONLY RATE BITS
2118 00003D80 80A7[B5380100]1F <1>      AND     byte [DSK_STATE+eDI], ~(RATE_MSK+DBL_STEP)
2119 <1> ; RATE, DBL STEP OFF
2120 00003D87 08A7[B5380100] <1>      OR      [DSK_STATE+eDI],AH ; TURN ON NEW RATE
2121 00003D8D C605[A8380100]00 <1>      MOV     byte [DSKETTE_STATUS],0 ; RESET STATUS FOR RETRY
2122 00003D94 F9 <1>      STC ; SET CARRY FOR RETRY
2123 00003D95 C3 <1>      RETn ; RETRY RETURN
2124 <1>
2125 <1> NO_RETRY:
2126 00003D96 F8 <1>      CLC ; CLEAR CARRY NO RETRY
2127 00003D97 C3 <1>      RETn ; NO RETRY RETURN
2128 <1>
2129 <1> ;-----
2130 <1> ; NUM_TRANS
2131 <1> ; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE
2132 <1> ; ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
2133 <1> ;
2134 <1> ; ON ENTRY: [BP+1] = TRACK
2135 <1> ; SI-HI = HEAD
2136 <1> ; [BP] = START SECTOR
2137 <1> ;
2138 <1> ; ON EXIT: AL = NUMBER ACTUALLY TRANSFERRED
2139 <1> ;-----
2140 <1> NUM_TRANS:
2141 00003D98 30C0 <1>      XOR     AL,AL ; CLEAR FOR ERROR
2142 00003D9A 803D[A8380100]00 <1>      CMP     byte [DSKETTE_STATUS],0 ; CHECK FOR ERROR
2143 00003DA1 752C <1>      JNZ     NT_OUT ; IF ERROR 0 TRANSFERRED
2144 00003DA3 B204 <1>      MOV     DL,4 ; SECTORS/TRACK OFFSET TO DL
2145 00003DA5 E8FA000000 <1>      CALL    GET_PARM ; AH = SECTORS/TRACK
2146 00003DAA 8A1D[AE380100] <1>      MOV     BL,[NEC_STATUS+5] ; GET ENDING SECTOR
2147 00003DB0 6689F1 <1>      MOV     CX,SI ; CH = HEAD # STARTED
2148 00003DB3 3A2D[AD380100] <1>      CMP     CH,[NEC_STATUS+4] ; GET HEAD ENDED UP ON
2149 00003DB9 750D <1>      JNZ     DIF_HD ; IF ON SAME HEAD, THEN NO ADJUST
2150 00003DBB 8A2D[AC380100] <1>      MOV     CH,[NEC_STATUS+3] ; GET TRACK ENDED UP ON
2151 00003DC1 3A6D01 <1>      CMP     CH,[eBP+1] ; IS IT ASKED FOR TRACK
2152 00003DC4 7404 <1>      JZ      short SAME_TRK ; IF SAME TRACK NO INCREASE
2153 00003DC6 00E3 <1>      ADD     BL,AH ; ADD SECTORS/TRACK
2154 <1> DIF_HD:
2155 00003DC8 00E3 <1>      ADD     BL,AH ; ADD SECTORS/TRACK
2156 <1> SAME_TRK:
2157 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[A7380100] <1> MOV [MOTOR_COUNT],AH ; STORE UPON RETURN
2175 00003DDF 6658 <1> POP AX ; RESTORE NUMBER TRANSFERRED
2176 00003DE1 8A25[A8380100] <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[B5380100] <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[A5380100]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[B5380100]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[A7380100]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[A8380100]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[AC380100] <1> MOV CL, [NEC_STATUS+3] ; LOAD RETURNED TRACK
2247 00003E68 888F[B9380100] <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[B5380100]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> ;
2260 <1> ; ON ENTRY: DI : BIT 2 = HEAD; BITS 1,0 = DRIVE
2261 <1> ;
2262 <1> ; ON EXIT: DI : BIT 2 IS RESET, BITS 1,0 = DRIVE
2263 <1> ; @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
2264 <1> ;-----
2265 <1> READ_ID:
2266 00003E7D B8[9A3E0000] <1> MOV eAX, ER_3 ; MOVE NEC OUTPUT ERROR ADDRESS
2267 00003E82 50 <1> PUSH eAX
2268 00003E83 B44A <1> MOV AH,4AH ; READ ID COMMAND

```



```

2269 00003E85 E820010000 <1> CALL NEC_OUTPUT ; TO CONTROLLER
2270 00003E8A 6689F8 <1> MOV AX,DI ; DRIVE # TO AH, HEAD 0
2271 00003E8D 88C4 <1> MOV AH,AL
2272 00003E8F E816010000 <1> CALL NEC_OUTPUT ; TO CONTROLLER
2273 00003E94 E80BFEFFFF <1> CALL NEC_TERM ; WAIT FOR OPERATION, GET STATUS
2274 00003E99 58 <1> POP eAX ; THROW AWAY ERROR ADDRESS
2275 <1> ER_3:
2276 00003E9A C3 <1> RETn
2277 <1>
2278 <1> ;-----
2279 <1> ; CMOS_TYPE
2280 <1> ; RETURNS DISKETTE TYPE FROM CMOS
2281 <1> ;
2282 <1> ; ON ENTRY: DI = DRIVE #
2283 <1> ;
2284 <1> ; ON EXIT: AL = TYPE; CY REFLECTS STATUS
2285 <1> ;-----
2286 <1>
2287 <1> CMOS_TYPE: ; 11/12/2014
2288 00003E9B 8A87[565C0000] <1> mov al, [eDI+fd0_type]
2289 00003EA1 20C0 <1> and al, al ; 18/12/2014
2290 00003EA3 C3 <1> retn
2291 <1>
2292 <1> ;CMOS_TYPE:
2293 <1> ; MOV AL, CMOS_DIAG ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
2294 <1> ; CALL CMOS_READ ; GET CMOS STATUS
2295 <1> ; TEST AL,BAD_BAT+BAD_CKSUM ; BATTERY GOOD AND CHECKSUM VALID
2296 <1> ; STC ; SET CY = 1 INDICATING ERROR FOR RETURN
2297 <1> ; JNZ short BAD_CM ; ERROR IF EITHER BIT ON
2298 <1> ; MOV AL,CMOS_DISKETTE ; ADDRESS OF DISKETTE BYTE IN CMOS
2299 <1> ; CALL CMOS_READ ; GET DISKETTE BYTE
2300 <1> ; OR DI,DI ; SEE WHICH DRIVE IN QUESTION
2301 <1> ; JNZ short TB ; IF DRIVE 1, DATA IN LOW NIBBLE
2302 <1> ; ROR AL,4 ; EXCHANGE NIBBLES IF SECOND DRIVE
2303 <1> ;TB:
2304 <1> ; AND AL,0FH ; KEEP ONLY DRIVE DATA, RESET CY, 0
2305 <1> ;BAD_CM:
2306 <1> ; RETn ; CY, STATUS OF READ
2307 <1>
2308 <1> ;-----
2309 <1> ; GET_PARM
2310 <1> ; THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK_BASE
2311 <1> ; BLOCK POINTED TO BY THE DATA VARIABLE @DISK_POINTER. A BYTE FROM
2312 <1> ; THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING
2313 <1> ; THE PARAMETER IN DL.
2314 <1> ;
2315 <1> ; ON ENTRY: DL = INDEX OF BYTE TO BE FETCHED
2316 <1> ;
2317 <1> ; ON EXIT: AH = THAT BYTE FROM BLOCK
2318 <1> ; AL,DH DESTROYED
2319 <1> ;-----
2320 <1> GET_PARM:
2321 <1> ;PUSH DS
2322 00003EA4 56 <1> PUSH eSI
2323 <1> ;SUB AX,AX ; DS = 0, BIOS DATA AREA
2324 <1> ;MOV DS,AX
2325 <1> ;imov ax, cs
2326 <1> ;imov ds, ax
2327 <1> ; 08/02/2015 (protected mode modifications, bx -> ebx)
2328 00003EA5 87D3 <1> XCHG eDX,eBX ; BL = INDEX
2329 <1> ;SUB BH,BH ; BX = INDEX
2330 00003EA7 81E3FF000000 <1> and ebx, 0FFh
2331 <1> ;LDS SI, [DISK_POINTER] ; POINT TO BLOCK
2332 <1> ;
2333 <1> ; 17/12/2014
2334 00003EAD 66A1[455C0000] <1> mov ax, [cfd] ; current (AL) and previous fd (AH)
2335 00003EB3 38E0 <1> cmp al, ah
2336 00003EB5 7425 <1> je short gpndc
2337 00003EB7 A2[465C0000] <1> mov [pfd], al ; current drive -> previous drive
2338 00003EBC 53 <1> push ebx ; 08/02/2015
2339 00003EBD 88C3 <1> mov bl, al
2340 <1> ; 11/12/2014
2341 00003EBF 8A83[565C0000] <1> mov al, [eBX+fd0_type] ; Drive type (0,1,2,3,4)
2342 <1> ; 18/12/2014
2343 00003EC5 20C0 <1> and al, al
2344 00003EC7 7507 <1> jnz short gpdtc
2345 00003EC9 BB[2F5C0000] <1> mov ebx, MD_TBL6 ; 1.44 MB param. tbl. (default)
2346 0000ECE EB05 <1> jmp short gpdpu
2347 <1> gpdtc:
2348 00003ED0 E817F9FFFF <1> call DR_TYPE_CHECK
2349 <1> ; cf = 1 -> eBX points to 1.44MB fd parameter table (default)
2350 <1> gpdpu:
2351 00003ED5 891D[CC5B0000] <1> mov [DISK_POINTER], ebx
2352 00003EDB 5B <1> pop ebx
2353 <1> gpndc:
2354 00003EDC 8B35[CC5B0000] <1> mov esi, [DISK_POINTER] ; 08/02/2015, si -> esi
2355 00003EE2 8A241E <1> MOV AH, [eSI+eBX] ; GET THE WORD
2356 00003EE5 87D3 <1> XCHG eDX,eBX ; RESTORE BX
2357 00003EE7 5E <1> POP eSI
2358 <1> ;POP DS
2359 00003EE8 C3 <1> RETn
2360 <1>
2361 <1> ;-----
2362 <1> ; MOTOR_ON
2363 <1> ; TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT
2364 <1> ; IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFh) TO ENSURE
2365 <1> ; THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
2366 <1> ; MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION
2367 <1> ; (AX=90FDH, INT 15) IS CALLED TELLING THE OPERATING SYSTEM
2368 <1> ; THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS
2369 <1> ; FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
2370 <1> ; HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE

```

```

2371 <1> ; THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
2372 <1> ; NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE
2373 <1> ; PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
2374 <1> ; IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
2375 <1> ; WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.
2376 <1> ;
2377 <1> ; ON ENTRY: DI = DRIVE #
2378 <1> ; ON EXIT: AX,CX,DX DESTROYED
2379 <1> ;-----
2380 <1> MOTOR_ON:
2381 <1> PUSH eBX ; SAVE REG.
2382 <1> CALL TURN_ON ; TURN ON MOTOR
2383 <1> JC short MOT_IS_ON ; IF CY=1 NO WAIT
2384 <1> CALL XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
2385 <1> CALL XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH,
2386 <1> ;CALL TURN_ON ; CHECK AGAIN IF MOTOR ON
2387 <1> ;JC MOT_IS_ON ; IF NO WAIT MEANS IT IS ON
2388 <1> M_WAIT:
2389 <1> MOV DL,10 ; GET THE MOTOR WAIT PARAMETER
2390 <1> CALL GET_PARM
2391 <1> ;MOV AL,AH ; AL = MOTOR WAIT PARAMETER
2392 <1> ;XOR AH,AH ; AX = MOTOR WAIT PARAMETER
2393 <1> ;CMP AL,8 ; SEE IF AT LEAST A SECOND IS SPECIFIED
2394 <1> cmp ah, 8
2395 <1> ;JAE short GP2 ; IF YES, CONTINUE
2396 <1> ja short J13
2397 <1> ;MOV AL,8 ; ONE SECOND WAIT FOR MOTOR START UP
2398 <1> mov ah, 8
2399 <1>
2400 <1> ;----- AS CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
2401 <1> GP2:
2402 <1> ;----- FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
2403 <1> J13: ; WAIT FOR 1/8 SECOND PER (AL)
2404 <1> MOV eCX,8286 ; COUNT FOR 1/8 SECOND AT 15.085737 US
2405 <1> CALL WAITF ; GO TO FIXED WAIT ROUTINE
2406 <1> ;DEC AL ; DECREMENT TIME VALUE
2407 <1> dec ah
2408 <1> JNZ short J13 ; ARE WE DONE YET
2409 <1> MOT_IS_ON:
2410 <1> POP eBX ; RESTORE REG.
2411 <1> RETn
2412 <1>
2413 <1> ;-----
2414 <1> ; TURN_ON
2415 <1> ; TURN MOTOR ON AND RETURN WAIT STATE.
2416 <1> ;
2417 <1> ; ON ENTRY: DI = DRIVE #
2418 <1> ;
2419 <1> ; ON EXIT: CY = 0 MEANS WAIT REQUIRED
2420 <1> ; CY = 1 MEANS NO WAIT REQUIRED
2421 <1> ; AX,BX,CX,DX DESTROYED
2422 <1> ;-----
2423 <1> TURN_ON:
2424 <1> MOV eBX,eDI ; BX = DRIVE #
2425 <1> MOV CL,BL ; CL = DRIVE #
2426 <1> ROL BL,4 ; BL = DRIVE SELECT
2427 <1> CLI ; NO INTERRUPTS WHILE DETERMINING STATUS
2428 <1> MOV byte [MOTOR_COUNT],0FFH ; ENSURE MOTOR STAYS ON FOR OPERATION
2429 <1> MOV AL, [MOTOR_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION
2430 <1> AND AL,00110000B ; KEEP ONLY DRIVE SELECT BITS
2431 <1> MOV AH,1 ; MASK FOR DETERMINING MOTOR BIT
2432 <1> SHL AH,CL ; AH = MOTOR ON, A=00000001, B=00000010
2433 <1>
2434 <1> ; AL = DRIVE SELECT FROM @MOTOR_STATUS
2435 <1> ; BL = DRIVE SELECT DESIRED
2436 <1> ; AH = MOTOR ON MASK DESIRED
2437 <1>
2438 <1> CMP AL,BL ; REQUESTED DRIVE ALREADY SELECTED ?
2439 <1> JNZ short TURN_IT_ON ; IF NOT SELECTED JUMP
2440 <1> TEST AH, [MOTOR_STATUS] ; TEST MOTOR ON BIT
2441 <1> JNZ short NO_MOT_WAIT ; JUMP IF MOTOR ON AND SELECTED
2442 <1>
2443 <1> TURN_IT_ON:
2444 <1> OR AH,BL ; AH = DRIVE SELECT AND MOTOR ON
2445 <1> MOV BH,[MOTOR_STATUS] ; SAVE COPY OF @MOTOR_STATUS BEFORE
2446 <1> AND BH,00001111B ; KEEP ONLY MOTOR BITS
2447 <1> AND byte [MOTOR_STATUS],11001111B ; CLEAR OUT DRIVE SELECT
2448 <1> OR [MOTOR_STATUS],AH ; OR IN DRIVE SELECTED AND MOTOR ON
2449 <1> MOV AL,[MOTOR_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION
2450 <1> MOV BL,AL ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
2451 <1> AND BL,00001111B ; KEEP ONLY MOTOR BITS
2452 <1> STI ; ENABLE INTERRUPTS AGAIN
2453 <1> AND AL,00111111B ; STRIP AWAY UNWANTED BITS
2454 <1> ROL AL,4 ; PUT BITS IN DESIRED POSITIONS
2455 <1> OR AL,00001100B ; NO RESET, ENABLE DMA/INTERRUPT
2456 <1> MOV DX,03F2H ; SELECT DRIVE AND TURN ON MOTOR
2457 <1> OUT DX,AL
2458 <1> CMP BL,BH ; NEW MOTOR TURNED ON ?
2459 <1> ;JZ short NO_MOT_WAIT ; NO WAIT REQUIRED IF JUST SELECT
2460 <1> je short no_mot_w1 ; 27/02/2015
2461 <1> CLC ; (re)SET CARRY MEANING WAIT
2462 <1> RETn
2463 <1>
2464 <1> NO_MOT_WAIT:
2465 <1> sti
2466 <1> no_mot_w1: ; 27/02/2015
2467 <1> STC ; SET NO WAIT REQUIRED
2468 <1> ;STI ; INTERRUPTS BACK ON
2469 <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 <1> MOV DL,9 ; GET HEAD SETTLE PARAMETER
2481 <1> CALL GET_PARM
2482 <1> or ah, ah ; 17/12/2014
2483 <1> jnz short DO_WAT
2484 <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 <1> jz short HW_DONE
2489 <1> MOV AH,HD12_SETTLE ; LOAD 1.2M HEAD SETTLE MINIMUM
2490 <1> MOV AL,[DSK_STATE+eDI] ; LOAD STATE
2491 <1> AND AL,RATE_MSK ; KEEP ONLY RATE
2492 <1> CMP AL,RATE_250 ; 1.2 M DRIVE ?
2493 <1> JNZ short DO_WAT ; DEFAULT HEAD SETTLE LOADED
2494 <1> ;GP3:
2495 <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 <1> MOV eCX,66 ; COUNT AT 15.085737 US PER COUNT
2509 <1> CALL WAITF ; DELAY FOR 1 MILLISECOND
2510 <1> ;DEC AL ; DECREMENT THE COUNT
2511 <1> dec ah
2512 <1> JNZ short J29 ; DO AL MILLISECOND # OF TIMES
2513 <1> HW_DONE:
2514 <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 microprocessor.
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, diskette drive 1 is in the seek mode.
2547 <1> ; Bit 0 - When this bit is set to 1, diskette 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 <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> ;jnb short J23
2617 <1>
2618 <1> ;WFPS_TIMEOUT:
2619 <1>
2620 <1> ;----- FALL THRU TO ERROR RETURN
2621 <1>
2622 00003FC8 800D[A8380100]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[A5380100] <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[A5380100] <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[A8380100]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[B9380100]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[B5380100]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[B9380100] <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[B9380100] <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[B9380100] <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[A9380100] <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[A8380100]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[A5380100]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[A8380100]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[A5380100]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[A9380100] <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:
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:
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[A8380100]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[A8380100]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[A9380100]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[B5380100]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[B5380100]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[A5380100]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> ; DISKETTE_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[B5380100]00- <1>      MOV     WORD [DSK_STATE],0 ; INITIALIZE STATES
3063 000041D3 00        <1>
3064 000041D4 8025[B0380100]33 <1>      AND     byte [LASTRATE],~(STRT_MSK+SEND_MSK) ; CLEAR START & SEND
3065 000041DB 800D[B0380100]C0 <1>      OR      byte [LASTRATE],SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
3066 000041E2 C605[A5380100]00 <1>      MOV     byte [SEEK_STATUS],0 ; INDICATE RECALIBRATE NEEDED
3067 000041E9 C605[A7380100]00 <1>      MOV     byte [MOTOR_COUNT],0 ; INITIALIZE MOTOR COUNT
3068 000041F0 C605[A6380100]00 <1>      MOV     byte [MOTOR_STATUS],0 ; INITIALIZE DRIVES TO OFF STATE
3069 000041F7 C605[A8380100]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 [fdl_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_CNTLRL 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
3378 <1> ; EQU 00001000B ; NOT USED
3379 <1> ERR_ID EQU 00010000B ; ID NOT FOUND
3380 <1> ; EQU 00100000B ; NOT USED
3381 <1> ERR_DATA_ECC EQU 01000000B
3382 <1> ERR_BAD_BLOCK EQU 10000000B
3383 <1>
3384 <1>
3385 <1> RECAL_CMD EQU 00010000B ; DRIVE RECAL(10H)
3386 <1> READ_CMD EQU 00100000B ; READ (20H)

```

```

3387 <1> WRITE_CMD EQU 00110000B ; WRITE (30H)
3388 <1> VERIFY_CMD EQU 01000000B ; VERIFY (40H)
3389 <1> FMTTRK_CMD EQU 01010000B ; FORMAT TRACK (50H)
3390 <1> INIT_CMD EQU 01100000B ; INITIALIZE (60H)
3391 <1> SEEK_CMD EQU 01110000B ; SEEK (70H)
3392 <1> DIAG_CMD EQU 10010000B ; DIAGNOSTIC (90H)
3393 <1> SET_PARM_CMD EQU 10010001B ; DRIVE PARMS (91H)
3394 <1> NO_RETRIES EQU 00000001B ; CHD MODIFIER (01H)
3395 <1> ECC_MODE EQU 00000010B ; CMD MODIFIER (02H)
3396 <1> BUFFER_MODE EQU 00001000B ; CMD MODIFIER (08H)
3397 <1>
3398 <1> ;MAX_FILE EQU 2
3399 <1> ;S_MAX_FILE EQU 2
3400 <1> MAX_FILE equ 4 ; 22/12/2014
3401 <1> S_MAX_FILE equ 4 ; 22/12/2014
3402 <1>
3403 <1> DELAY_1 EQU 25H ; DELAY FOR OPERATION COMPLETE
3404 <1> DELAY_2 EQU 0600H ; DELAY FOR READY
3405 <1> DELAY_3 EQU 0100H ; DELAY FOR DATA REQUEST
3406 <1>
3407 <1> HF_FAIL EQU 08H ; CMOS FLAG IN BYTE 0EH
3408 <1>
3409 <1> ;----- COMMAND BLOCK REFERENCE
3410 <1>
3411 <1> ;CMD_BLOCK EQU BP-8 ; @CMD_BLOCK REFERENCES BLOCK HEAD IN SS
3412 <1> ; (BP) POINTS TO COMMAND BLOCK TAIL
3413 <1> ; AS DEFINED BY THE "ENTER" PARMS
3414 <1> ; 19/12/2014
3415 <1> ORG_VECTOR equ 4*13h ; INT 13h vector
3416 <1> DISK_VECTOR equ 4*40h ; INT 40h vector (for floppy disks)
3417 <1> ;HDISK_INT equ 4*76h ; Primary HDC - Hardware interrupt (IRQ14)
3418 <1> ;HDISK_INT1 equ 4*76h ; Primary HDC - Hardware interrupt (IRQ14)
3419 <1> ;HDISK_INT2 equ 4*77h ; Secondary HDC - Hardware interrupt (IRQ15)
3420 <1> ;HF_TBL_VEC equ 4*41h ; Pointer to 1st fixed disk parameter table
3421 <1> ;HF1_TBL_VEC equ 4*46h ; Pointer to 2nd fixed disk parameter table
3422 <1>
3423 <1> align 2
3424 <1>
3425 <1> ;-----
3426 <1> ; FIXED DISK I/O SETUP :
3427 <1> ; :
3428 <1> ; - ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK :
3429 <1> ; - PERFORM POWER ON DIAGNOSTICS :
3430 <1> ; SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED :
3431 <1> ; :
3432 <1> ;-----
3433 <1>
3434 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
3435 <1>
3436 <1> DISK_SETUP:
3437 <1> ;CLI
3438 <1> ;;MOV AX,ABS0 ; GET ABSOLUTE SEGMENT
3439 <1> ;xor ax,ax
3440 <1> ;MOV DS,AX ; SET SEGMENT REGISTER
3441 <1> ;MOV AX,[ORG_VECTOR] ; GET DISKETTE VECTOR
3442 <1> ;MOV [DISK_VECTOR],AX ; INTO INT 40H
3443 <1> ;MOV AX,[ORG_VECTOR+2]
3444 <1> ;MOV [DISK_VECTOR+2],AX
3445 <1> ;MOV word [ORG_VECTOR],DISK_IO ; FIXED DISK HANDLER
3446 <1> ;MOV [ORG_VECTOR+2],CS
3447 <1> ; 1st controller (primary master, slave) - IRQ 14
3448 <1> ;;MOV word [HDISK_INT],HD_INT ; FIXED DISK INTERRUPT
3449 <1> ;mov word [HDISK_INT1],HD_INT ;
3450 <1> ;;MOV [HDISK_INT+2],CS
3451 <1> ;mov [HDISK_INT1+2],CS
3452 <1> ; 2nd controller (secondary master, slave) - IRQ 15
3453 <1> ;mov word [HDISK_INT2],HD1_INT ;
3454 <1> ;mov [HDISK_INT2+2],CS
3455 <1> ;
3456 <1> ;;MOV word [HF_TBL_VEC],HD0_DPT ; PARM TABLE DRIVE 80
3457 <1> ;;MOV word [HF_TBL_VEC+2],DPT_SEGM
3458 <1> ;;MOV word [HF1_TBL_VEC],HD1_DPT ; PARM TABLE DRIVE 81
3459 <1> ;;MOV word [HF1_TBL_VEC+2],DPT_SEGM
3460 <1> ;push cs
3461 <1> ;pop ds
3462 <1> ;mov word [HDPM_TBL_VEC],HD0_DPT ; PARM TABLE DRIVE 80h
3463 <1> ;mov word [HDPM_TBL_VEC+2],DPT_SEGM
3464 0000420E C705[C0380100]0000- <1> mov dword [HDPM_TBL_VEC], (DPT_SEGM*16)+HD0_DPT
3464 00004216 0900 <1>
3465 <1> ;mov word [HDPS_TBL_VEC],HD1_DPT ; PARM TABLE DRIVE 81h
3466 <1> ;mov word [HDPS_TBL_VEC+2],DPT_SEGM
3467 00004218 C705[C4380100]2000- <1> mov dword [HDPS_TBL_VEC], (DPT_SEGM*16)+HD1_DPT
3467 00004220 0900 <1>
3468 <1> ;mov word [HDSM_TBL_VEC],HD2_DPT ; PARM TABLE DRIVE 82h
3469 <1> ;mov word [HDSM_TBL_VEC+2],DPT_SEGM
3470 00004222 C705[C8380100]4000- <1> mov dword [HDSM_TBL_VEC], (DPT_SEGM*16)+HD2_DPT
3470 0000422A 0900 <1>
3471 <1> ;mov word [HDSS_TBL_VEC],HD3_DPT ; PARM TABLE DRIVE 83h
3472 <1> ;mov word [HDSS_TBL_VEC+2],DPT_SEGM
3473 0000422C C705[CC380100]6000- <1> mov dword [HDSS_TBL_VEC], (DPT_SEGM*16)+HD3_DPT
3473 00004234 0900 <1>
3474 <1> ;
3475 <1> ;;IN AL,INTB01 ; TURN ON SECOND INTERRUPT CHIP
3476 <1> ;;;AND AL,0BFH
3477 <1> ;;and al,3Fh ; enable IRQ 14 and IRQ 15
3478 <1> ;;;JMP $+2
3479 <1> ;;IODELAY
3480 <1> ;;OUT INTB01,AL
3481 <1> ;;IODELAY
3482 <1> ;;IN AL,INTA01 ; LET INTERRUPTS PASS THRU TO
3483 <1> ;;AND AL,0FBH ; SECOND CHIP
3484 <1> ;;;JMP $+2

```



```

3485 <1> ; IODELAY
3486 <1> ; OUT INTA01,AL
3487 <1> ;
3488 <1> ; STI
3489 <1> ; PUSH DS ; MOVE ABS0 POINTER TO
3490 <1> ; POP ES ; EXTRA SEGMENT POINTER
3491 <1> ; ;CALL DDS ; ESTABLISH DATA SEGMENT
3492 <1> ; MOV byte [DISK_STATUS1],0 ; RESET THE STATUS INDICATOR
3493 <1> ; MOV byte [HF_NUM],0 ; ZERO NUMBER OF FIXED DISKS
3494 <1> ; MOV byte [CONTROL_BYTE],0
3495 <1> ; MOV byte [PORT_OFF],0 ; ZERO CARD OFFSET
3496 <1> ; 20/12/2014 - private code by Erdogan Tan
3497 <1> ; (out of original PC-AT, PC-XT BIOS code)
3498 <1> ; mov si, hd0_type
3499 00004236 BE[585C0000] <1> mov esi, hd0_type
3500 <1> ; mov cx, 4
3501 0000423B B904000000 <1> mov ecx, 4
3502 <1> hde_l:
3503 00004240 AC <1> lodsb
3504 00004241 3C80 <1> cmp al, 80h ; 8?h = existing
3505 00004243 7206 <1> jnb short _L4
3506 00004245 FE05[BC380100] <1> inc byte [HF_NUM] ; + 1 hard (fixed) disk drives
3507 <1> _L4: ; 26/02/2015
3508 0000424B E2F3 <1> loop hde_l
3509 <1> ; _L4: ; 0 <= [HF_NUM] =< 4
3510 <1> ; _L4:
3511 <1> ;
3512 <1> ; ; 31/12/2014 - cancel controller diagnostics here
3513 <1> ; ; mov cx, 3 ; 26/12/2014 (Award BIOS 1999)
3514 <1> ; ; mov cl, 3
3515 <1> ; ;
3516 <1> ; ; MOV DL,80H ; CHECK THE CONTROLLER
3517 <1> ; ; hdc_dl:
3518 <1> ; ; MOV AH,14H ; USE CONTROLLER DIAGNOSTIC COMMAND
3519 <1> ; ; INT 13H ; CALL BIOS WITH DIAGNOSTIC COMMAND
3520 <1> ; ; JC short CTL_ERRX ; DISPLAY ERROR MESSAGE IF BAD RETURN
3521 <1> ; ; jc short POD_DONE ; 22/12/2014
3522 <1> ; ; jnc short hdc_reset0
3523 <1> ; ; loop hdc_dl
3524 <1> ; ; ; 27/12/2014
3525 <1> ; ; stc
3526 <1> ; ; retn
3527 <1> ;
3528 <1> ; ; hdc_reset0:
3529 <1> ; ; 18/01/2015
3530 0000424D 8A0D[BC380100] <1> mov cl, [HF_NUM]
3531 00004253 20C9 <1> and cl, cl
3532 00004255 740E <1> jz short POD_DONE
3533 <1> ;
3534 00004257 B27F <1> mov dl, 7Fh
3535 <1> hdc_reset1:
3536 00004259 FEC2 <1> inc dl
3537 <1> ; ; 31/12/2015
3538 <1> ; ; push dx
3539 <1> ; ; push cx
3540 <1> ; ; push ds
3541 <1> ; ; sub ax, ax
3542 <1> ; ; mov ds, ax
3543 <1> ; ; MOV AX, [TIMER_LOW] ; GET START TIMER COUNTS
3544 <1> ; ; pop ds
3545 <1> ; ; MOV BX,AX
3546 <1> ; ; ADD AX,6*182 ; 60 SECONDS* 18.2
3547 <1> ; ; MOV CX,AX
3548 <1> ; ; mov word [wait_count], 0 ; 22/12/2014 (reset wait counter)
3549 <1> ; ;
3550 <1> ; ; 31/12/2014 - cancel HD_RESET_1
3551 <1> ; ; CALL HD_RESET_1 ; SET UP DRIVE 0, (1,2,3)
3552 <1> ; ; pop cx
3553 <1> ; ; pop dx
3554 <1> ; ;
3555 <1> ; ; 18/01/2015
3556 0000425B B40D <1> mov ah, 0Dh ; ALTERNATE RESET
3557 <1> ; ; int 13h
3558 0000425D E8A4FFFFFF <1> call int13h
3559 00004262 E2F5 <1> loop hdc_reset1
3560 00004264 F8 <1> clc ; 29/05/2016
3561 <1> POD_DONE:
3562 00004265 C3 <1> RETn
3563 <1>
3564 <1> ; ; ---- POD_ERROR
3565 <1>
3566 <1> ; ; CTL_ERRX:
3567 <1> ; ; MOV SI,OFFSET F1782 ; CONTROLLER ERROR
3568 <1> ; ; CALL SET_FAIL ; DO NOT IPL FROM DISK
3569 <1> ; ; CALL E_MSG ; DISPLAY ERROR AND SET (BP) ERROR FLAG
3570 <1> ; ; JMP short POD_DONE
3571 <1>
3572 <1> ; ; HD_RESET_1:
3573 <1> ; ; ; PUSH BX ; SAVE TIMER LIMITS
3574 <1> ; ; ; PUSH CX
3575 <1> ; ; RES_1: MOV AH,09H ; SET DRIVE PARAMETERS
3576 <1> ; ; INT 13H
3577 <1> ; ; JC short RES_2
3578 <1> ; ; MOV AH,11H ; RECALIBRATE DRIVE
3579 <1> ; ; INT 13H
3580 <1> ; ; JNC short RES_CHK ; 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> ; ; ; jnb 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_CHK: 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>
3680 <1> ;-----
3681 <1> ; FIXED DISK BIOS ENTRY POINT :
3682 <1> ;-----
3683 <1>
3684 <1> ; 15/01/2017
3685 <1> ; 14/01/2017
3686 <1> ; 07/01/2017
3687 <1> ; 02/01/2017
3688 <1> ; 01/06/2016

```

```

3689 <1> ; 16/05/2016, 27/05/2016, 28/05/2016, 29/05/2016
3690 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
3691 <1> int33h: ; DISK I/O
3692 <1> ; 29/05/2016
3693 00004266 80642408FE <1> and byte [esp+8], 11111110b ; clear carry bit of eflags register
3694 <1> ; 16/05/2016
3695 0000426B 1E <1> push ds
3696 0000426C 53 <1> push ebx ; user's buffer address (virtual)
3697 0000426D 66BB1000 <1> mov bx, KDATA ; System (Kernel's) data segment
3698 00004271 8EDB <1> mov ds, bx
3699 <1>
3700 <1> ;;15/01/2017
3701 <1> ; 14/01/2017
3702 <1> ; 02/01/2017
3703 <1> ;mov byte [intflg], 33h ; disk io interrupt
3704 <1> ;pop ebx
3705 <1> ;mov [user_buffer], ebx
3706 <1>
3707 00004273 8F05[B0450100] <1> pop dword [user_buffer] ; 01/06/2016
3708 <1>
3709 00004279 C605[E63E0100]00 <1> mov byte [scount], 0 ; sector count for transfer
3710 00004280 80FC03 <1> cmp ah, 03h ; chs write
3711 00004283 7744 <1> ja short int33h_2
3712 00004285 7407 <1> je short int33h_0
3713 00004287 80FC02 <1> cmp ah, 02h ; chs read
3714 0000428A 726A <1> jb short int33h_5
3715 0000428C EB63 <1> jmp short int33h_4
3716 <1> int33h_0:
3717 <1> ; transfer user's buffer content to sector buffer
3718 0000428E 51 <1> push ecx
3719 0000428F 0FB6C8 <1> movzx ecx, al
3720 <1> int33h_1:
3721 00004292 56 <1> push esi
3722 00004293 8B35[B0450100] <1> mov esi, [user_buffer]
3723 <1> ; esi = user's buffer address (virtual, ebx)
3724 00004299 57 <1> push edi
3725 0000429A 06 <1> push es
3726 0000429B 50 <1> push eax
3727 0000429C 66B81000 <1> mov ax, KDATA
3728 000042A0 8EC0 <1> mov es, ax
3729 000042A2 BF00000700 <1> mov edi, Cluster_Buffer
3730 000042A7 C1E109 <1> shl ecx, 9 ; * 512
3731 000042AA E86BA50000 <1> call transfer_from_user_buffer
3732 000042AF 58 <1> pop eax
3733 000042B0 07 <1> pop es
3734 000042B1 5F <1> pop edi
3735 000042B2 5E <1> pop esi
3736 000042B3 59 <1> pop ecx
3737 000042B4 7340 <1> jnc short int33h_5
3738 000042B6 8B1D[B0450100] <1> mov ebx, [user_buffer] ; 01/06/2016
3739 000042BC 1F <1> pop ds
3740 <1>
3741 <1> ;;15/01/2017
3742 <1> ; 02/01/2017
3743 <1> ;cli
3744 <1> ;mov byte [ss:intflg], 0 ; 07/01/2017
3745 <1> ;
3746 <1> ; (*) 29/05/2016
3747 <1> ; (*) retf 4 ; skip eflags on stack
3748 <1>
3749 <1> ; 29/05/2016 -set carry flag on stack-
3750 <1> ; [esp] = EIP
3751 <1> ; [esp+4] = CS
3752 <1> ; [esp+8] = E-FLAGS
3753 000042BD 804C240801 <1> or byte [esp+8], 1 ; set carry bit of eflags register
3754 <1> ; [esp+12] = ESP (user)
3755 <1> ; [esp+16] = SS (User)
3756 000042C2 B8FF000000 <1> mov eax, 0FFh ; Unknown error !?
3757 <1> ;iretd
3758 000042C7 EB79 <1> jmp short int33h_7 ; 07/01/2017
3759 <1>
3760 <1> ; (*) 29/05/2016 - 'ref 4' intruction causes to stack fault
3761 <1> ; (OUTER-PRIVILEGE-LEVEL)
3762 <1> ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986
3763 <1> ; // RETF instruction:
3764 <1> ;
3765 <1> ; IF OperandMode=32 THEN
3766 <1> ; Load CS:EIP from stack;
3767 <1> ; Set CS RPL to CPL;
3768 <1> ; Increment eSP by 8 plus the immediate offset if it exists;
3769 <1> ; Load SS:eSP from stack;
3770 <1> ; ELSE (* OperandMode=16 *)
3771 <1> ; Load CS:IP from stack;
3772 <1> ; Set CS RPL to CPL;
3773 <1> ; Increment eSP by 4 plus the immediate offset if it exists;
3774 <1> ; Load SS:eSP from stack;
3775 <1> ; FI;
3776 <1> ;
3777 <1> ; //
3778 <1>
3779 <1> int33h_2:
3780 000042C9 80FC05 <1> cmp ah, 05h ; format track
3781 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[B0450100] <1>      mov     ebx, [user_buffer] ; user's buffer address
3796 000042EF EB0A      <1>      jmp     short int33h_6
3797                    <1> int33h_4:
3798 000042F1 A2[E63E0100] <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[B0450100] <1>      mov     ebx, [CS:user_buffer] ; 01/06/2016
3809 0000430A 723D      <1>      jc      short int33h_9
3810                    <1>      ;
3811 0000430C 2E803D[E63E0100]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[E63E0100] <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 E896A40000 <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>      jnb     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[BB380100] <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[BB380100] <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[BB380100] <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[BB380100]100 <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[BC380100] <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[D0380100] <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 sul ; no
4038 <1> ;inc byte [LBAMode]
4039 <1> ;sul:
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[485C0000] <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[4A5C0000] <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 sul
4059 000044B0 FE05[D0380100] <1> inc byte [LBAMode] ; 1
4060 <1> sul:
4061 000044B6 C0E804 <1> shr al, 4
4062 000044B9 2401 <1> and al, 1
4063 000044BB A2[4C5C0000] <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[BD380100] <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[BD380100] <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
4088 <1> ;pop es ; **
4089 000044DD 6658 <1> pop ax ; ***
4090 <1> ;;pop bx
4091 000044DF E9C6000000 <1> jmp DISK_RESET
4092 <1> su2:
4093 000044E4 803D[D0380100]100 <1> cmp byte [LBAMode], 0
4094 000044EB 7661 <1> jna short su3
4095 <1> ;
4096 <1> ; 02/02/2015 (LBA read/write function calls)

```

```

4097 000044ED 80FC1B      <1>      cmp     ah, 1Bh
4098 000044F0 720B      <1>      jnb     short lbarw1
4099 000044F2 80FC1C      <1>      cmp     ah, 1Ch
4100 000044F5 775C      <1>      ja      short invldfnc
4101                                <1>      ; ;pop dx ; * ; 14/02/2015
4102                                <1>      ;mov ax, cx ; Lower word of LBA address (bits 0-15)
4103 000044F7 89C8      <1>      mov     eax, ecx ; LBA address (21/02/2015)
4104                                <1>      ; ; 14/02/2015
4105 000044F9 88D1      <1>      mov     cl, dl ; 14/02/2015
4106                                <1>      ; ;mov dx, bx
4107                                <1>      ;mov dx, si ; higher word of LBA address (bits 16-23)
4108                                <1>      ; ;mov bx, di
4109                                <1>      ;mov si, di ; Buffer offset
4110 000044FB EB31      <1>      jmp     short lbarw2
4111                                <1>      lbarw1:
4112                                <1>      ; convert CHS to LBA
4113                                <1>      ;
4114                                <1>      ; LBA calculation - AWARD BIOS - 1999 - AHDSK.ASM
4115                                <1>      ; LBA = "# of Heads" * Sectors/Track * Cylinder + Head * Sectors/Track
4116                                <1>      ; + Sector - 1
4117 000044FD 6652      <1>      push    dx ; * ; ; 14/02/2015
4118                                <1>      ;xor dh, dh
4119 000044FF 31D2      <1>      xor     edx, edx
4120                                <1>      ;mov dl, [ES:BX+14] ; sectors per track (logical)
4121 00004501 8A530E      <1>      mov     dl, [ebx+14]
4122                                <1>      ;xor ah, ah
4123 00004504 31C0      <1>      xor     eax, eax
4124                                <1>      ;mov al, [ES:BX+2]; heads (logical)
4125 00004506 8A4302      <1>      mov     al, [ebx+2]
4126 00004509 FEC8      <1>      dec     al
4127 0000450B 6640      <1>      inc     ax ; 0 = 256
4128 0000450D 66F7E2      <1>      mul     dx
4129                                <1>      ; AX = # of Heads" * Sectors/Track
4130 00004510 6689CA      <1>      mov     dx, cx
4131                                <1>      ;and cx, 3Fh ; sector (1 to 63)
4132 00004513 83E13F      <1>      and     ecx, 3fh
4133 00004516 86D6      <1>      xchg    dl, dh
4134 00004518 C0EE06      <1>      shr     dh, 6
4135                                <1>      ; DX = cylinder (0 to 1023)
4136                                <1>      ;mul dx
4137                                <1>      ; DX:AX = # of Heads" * Sectors/Track * Cylinder
4138 0000451B F7E2      <1>      mul     edx
4139 0000451D FEC9      <1>      dec     cl ; sector - 1
4140                                <1>      ;add ax, cx
4141                                <1>      ;adc dx, 0
4142                                <1>      ; DX:AX = # of Heads" * Sectors/Track * Cylinder + Sector -1
4143 0000451F 01C8      <1>      add     eax, ecx
4144 00004521 6659      <1>      pop     cx ; * ; ch = head, cl = drive number (zero based)
4145                                <1>      ;push dx
4146                                <1>      ;push ax
4147 00004523 50      <1>      push    eax
4148                                <1>      ;mov al, [ES:BX+14] ; sectors per track (logical)
4149 00004524 8A430E      <1>      mov     al, [ebx+14]
4150 00004527 F6E5      <1>      mul     ch
4151                                <1>      ; AX = Head * Sectors/Track
4152 00004529 6699      <1>      cwd
4153                                <1>      ;pop dx
4154 0000452B 5A      <1>      pop     edx
4155                                <1>      ;add ax, dx
4156                                <1>      ;pop dx
4157                                <1>      ;adc dx, 0 ; add carry bit
4158 0000452C 01D0      <1>      add     eax, edx
4159                                <1>      lbarw2:
4160 0000452E 29D2      <1>      sub     edx, edx ; 21/02/2015
4161 00004530 88CA      <1>      mov     dl, cl ; 21/02/2015
4162 00004532 C645F800      <1>      mov     byte [CMD_BLOCK], 0 ; Features Register
4163                                <1>      ; NOTE: Features register (1F1h, 171h)
4164                                <1>      ; is not used for ATA device R/W functions.
4165                                <1>      ; It is old/obsolete 'write precompensation'
4166                                <1>      ; register and error register
4167                                <1>      ; for old ATA/IDE devices.
4168                                <1>      ; 18/01/2014
4169                                <1>      ;mov ch, [hf_m_s] ; Drive 0 (master) or 1 (slave)
4170 00004536 8A0D[4C5C0000] <1>      mov     cl, [hf_m_s]
4171                                <1>      ;shl ch, 4 ; bit 4 (drive bit)
4172                                <1>      ;or ch, 0E0h ; bit 5 = 1
4173                                <1>      ; ; bit 6 = 1 = LBA mode
4174                                <1>      ; ; bit 7 = 1
4175 0000453C 80C90E      <1>      or      cl, 0Eh ; 1110b
4176                                <1>      ;and dh, 0Fh ; LBA byte 4 (bits 24 to 27)
4177 0000453F 25FFFFFFF0F <1>      and     eax, 0FFFFFFFh
4178 00004544 C1E11C      <1>      shl     ecx, 28 ; 21/02/2015
4179                                <1>      ;or dh, ch
4180 00004547 09C8      <1>      or      eax, ecx
4181                                <1>      ; ;mov [CMD_BLOCK+2], al ; LBA byte 1 (bits 0 to 7)
4182                                <1>      ; ; (Sector Number Register)
4183                                <1>      ; ;mov [CMD_BLOCK+3], ah ; LBA byte 2 (bits 8 to 15)
4184                                <1>      ; ; (Cylinder Low Register)
4185                                <1>      ;mov [CMD_BLOCK+2], ax ; LBA byte 1, 2
4186                                <1>      ;mov [CMD_BLOCK+4], dl ; LBA byte 3 (bits 16 to 23)
4187                                <1>      ; ; (Cylinder High Register)
4188                                <1>      ; ;mov [CMD_BLOCK+5], dh ; LBA byte 4 (bits 24 to 27)
4189                                <1>      ; ; (Drive/Head Register)
4190                                <1>
4191                                <1>      ;mov [CMD_BLOCK+4], dx ; LBA byte 4, LBA & DEV select bits
4192 00004549 8945FA      <1>      mov     [CMD_BLOCK+2], eax ; 21/02/2015
4193                                <1>      ;14/02/2015
4194                                <1>      ;mov dl, cl ; Drive number (INIT_DRV)
4195 0000454C EB38      <1>      jmp     short su4
4196                                <1>      su3:
4197                                <1>      ; 02/02/2015
4198                                <1>      ; (Temporary functions 1Bh & 1Ch are not valid for CHS mode)

```

```

4199 0000454E 80FC14      <1>      cmp     ah, 14h
4200 00004551 7604      <1>      jna     short chsfnc
4201                    <1> invldfnc:
4202                    <1>      ; 14/02/2015
4203                    <1>      ;pop  es ; **
4204 00004553 6658      <1>      pop     ax ; ***
4205                    <1>      ;jmp     short BAD_COMMAND_POP
4206 00004555 EB49      <1>      jmp     short BAD_COMMAND
4207                    <1> chsfnc:
4208                    <1>      ;MOV  AX,[ES:BX+5]          ; GET WRITE PRE-COMPENSATION CYLINDER
4209 00004557 668B4305    <1>      mov     ax, [ebx+5]
4210 0000455B 66C1E802    <1>      SHR     AX,2
4211 0000455F 8845F8      <1>      MOV     [CMD_BLOCK],AL
4212                    <1>      ;;MOV  AL,[ES:BX+8]          ; GET CONTROL BYTE MODIFIER
4213                    <1>      ;;PUSH DX
4214                    <1>      ;;MOV  DX,[HF_REG_PORT]
4215                    <1>      ;;OUT  DX,AL              ; SET EXTRA HEAD OPTION
4216                    <1>      ;;POP  DX ; *
4217                    <1>      ;;POP  ES ; **
4218                    <1>      ;;MOV  AH,[CONTROL_BYTE]    ; SET EXTRA HEAD OPTION IN
4219                    <1>      ;;AND  AH,0C0H              ; CONTROL BYTE
4220                    <1>      ;;OR   AH,AL
4221                    <1>      ;;MOV  [CONTROL_BYTE],AH
4222                    <1>      ;
4223 00004562 88C8      <1>      MOV     AL,CL          ; GET SECTOR NUMBER
4224 00004564 243F      <1>      AND     AL,3FH
4225 00004566 8845FA      <1>      MOV     [CMD_BLOCK+2],AL
4226 00004569 886DFB      <1>      MOV     [CMD_BLOCK+3],CH    ; GET CYLINDER NUMBER
4227 0000456C 88C8      <1>      MOV     AL,CL
4228 0000456E C0E806      <1>      SHR     AL,6
4229 00004571 8845FC      <1>      MOV     [CMD_BLOCK+4],AL    ; CYLINDER HIGH ORDER 2 BITS
4230                    <1>      ;;05/01/2015
4231                    <1>      ;;MOV  AL,DL              ; DRIVE NUMBER
4232 00004574 A0[4C5C0000]    <1>      mov     al, [hf_m_s]
4233 00004579 C0E004      <1>      SHL     AL,4
4234 0000457C 80E60F      <1>      AND     DH,0FH          ; HEAD NUMBER
4235 0000457F 08F0      <1>      OR      AL,DH
4236                    <1>      ;OR   AL,80H or 20H
4237 00004581 0CA0      <1>      OR      AL,80h+20h      ; ECC AND 512 BYTE SECTORS
4238 00004583 8845FD      <1>      MOV     [CMD_BLOCK+5],AL    ; ECC/SIZE/DRIVE/HEAD
4239                    <1> su4:
4240                    <1>      ;POP  ES ; **
4241                    <1>      ;; 14/02/2015
4242                    <1>      ;;POP  AX
4243                    <1>      ;;MOV  [CMD_BLOCK+1],AL        ; SECTOR COUNT
4244                    <1>      ;;PUSH AX
4245                    <1>      ;;MOV  AL,AH              ; GET INTO LOW BYTE
4246                    <1>      ;;XOR  AH,AH              ; ZERO HIGH BYTE
4247                    <1>      ;;SAL  AX,1              ; *2 FOR TABLE LOOKUP
4248 00004586 6658      <1>      pop     ax ; ***
4249 00004588 8845F9      <1>      mov     [CMD_BLOCK+1], al
4250 0000458B 29DB      <1>      sub     ebx, ebx
4251 0000458D 88E3      <1>      mov     bl, ah
4252                    <1>      ;xor   bh, bh
4253                    <1>      ;sal   bx, 1
4254 0000458F 66C1E302    <1>      sal     bx, 2 ; 32 bit offset (21/02/2015)
4255                    <1>      ;;MOV  SI,AX              ; PUT INTO SI FOR BRANCH
4256                    <1>      ;;CMP  AX,D1L              ; TEST WITHIN RANGE
4257                    <1>      ;;JNB  short BAD_COMMAND_POP
4258                    <1>      ;cmp    bx, D1L
4259 00004593 83FB74      <1>      cmp     ebx, D1L
4260 00004596 7308      <1>      jnb     short BAD_COMMAND
4261                    <1>      ;xchg  bx, si
4262 00004598 87DE      <1>      xchgebx, esi
4263                    <1>      ;;;POP AX          ; RESTORE AX
4264                    <1>      ;;;POP BX          ; AND DATA ADDRESS
4265                    <1>
4266                    <1>      ;;PUSH CX
4267                    <1>      ;;PUSH AX          ; ADJUST ES:BX
4268                    <1>      ;MOV  CX,BX          ; GET 3 HIGH ORDER NIBBLES OF BX
4269                    <1>      ;SHR  CX,4
4270                    <1>      ;MOV  AX,ES
4271                    <1>      ;ADD  AX,CX
4272                    <1>      ;MOV  ES,AX
4273                    <1>      ;AND  BX,000FH          ; ES:BX CHANGED TO ES:000X
4274                    <1>      ;;POP  AX
4275                    <1>      ;;POP  CX
4276                    <1>      ;;JMP  word [CS:SI+D1]
4277                    <1>      ;jmp   word [SI+D1]
4278 0000459A FFA6[E8430000] <1>      jmp     dword [esi+D1]
4279                    <1> ;;BAD_COMMAND_POP:
4280                    <1> ;; POP  AX
4281                    <1> ;; POP  BX
4282                    <1> BAD_COMMAND:
4283 000045A0 C605[BB380100]01 <1>      MOV     byte [DISK_STATUS1],BAD_CMD ; COMMAND ERROR
4284 000045A7 B000      <1>      MOV     AL,0
4285 000045A9 C3        <1>      RETn
4286                    <1>
4287                    <1> ;-----
4288                    <1> ; RESET THE DISK SYSTEM (AH=00H) :
4289                    <1> ;-----
4290                    <1>
4291                    <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[4A5C0000] <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[BD380100] <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[485C0000] <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[BB380100]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[4C5C0000] <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[BB380100]00 <1>      MOV     byte [DISK_STATUS1],0 ; IGNORE ANY SET UP ERRORS
4382 0000461A C3        <1>      RETn
4383                    <1>  ;drst6:
4384                    <1>      ; 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>  ;-----
4392                    <1>
4393                    <1>  RETURN_STATUS:
4394 00004621 A0[BB380100] <1>      MOV     AL,[DISK_STATUS1] ; OBTAIN PREVIOUS STATUS
4395 00004626 C605[BB380100]00 <1>      MOV     byte [DISK_STATUS1],0 ; RESET STATUS
4396 0000462D C3        <1>      RETn
4397                    <1>
4398                    <1>  ;-----
4399                    <1>      ; DISK READ ROUTINE (AH = 02H) :
4400                    <1>  ;-----

```

```

4401 <1>
4402 <1> DISK_READ:
4403 <1>     MOV     byte [CMD_BLOCK+6],READ_CMD
4404 <1>     JMP     COMMANDI
4405 <1>
4406 <1> ;-----
4407 <1> ;     DISK WRITE ROUTINE     (AH = 03H) :
4408 <1> ;-----
4409 <1>
4410 <1> DISK_WRITE:
4411 <1>     MOV     byte [CMD_BLOCK+6],WRITE_CMD
4412 <1>     JMP     COMMANDO
4413 <1>
4414 <1> ;-----
4415 <1> ;     DISK VERIFY           (AH = 04H) :
4416 <1> ;-----
4417 <1>
4418 <1> DISK_VERF:
4419 <1>     MOV     byte [CMD_BLOCK+6],VERIFY_CMD
4420 <1>     CALL    COMMAND
4421 <1>     JNZ     short VERF_EXIT           ; CONTROLLER STILL BUSY
4422 <1>     CALL    _WAIT                     ; (Original: CALL WAIT)
4423 <1>     JNZ     short VERF_EXIT           ; TIME OUT
4424 <1>     CALL    CHECK_STATUS
4425 <1> VERF_EXIT:
4426 <1>     RETn
4427 <1>
4428 <1> ;-----
4429 <1> ;     FORMATTING             (AH = 05H) :
4430 <1> ;-----
4431 <1>
4432 <1> FMT_TRK:           ; FORMAT TRACK     (AH = 005H)
4433 <1>     MOV     byte [CMD_BLOCK+6],FMTTRK_CMD
4434 <1>     ;PUSH   ES
4435 <1>     ;PUSH   BX
4436 <1>     push    ebx
4437 <1>     CALL    GET_VEC                 ; GET DISK PARAMETERS ADDRESS
4438 <1>     ;MOV    AL,[ES:BX+14]           ; GET SECTORS/TRACK
4439 <1>     mov     al, [ebx+14]
4440 <1>     MOV     [CMD_BLOCK+1],AL       ; SET SECTOR COUNT IN COMMAND
4441 <1>     pop     ebx
4442 <1>     ;POP    BX
4443 <1>     ;POP    ES
4444 <1>     JMP     CMD_OF                 ; GO EXECUTE THE COMMAND
4445 <1>
4446 <1> ;-----
4447 <1> ;     READ DASD TYPE         (AH = 15H) :
4448 <1> ;-----
4449 <1>
4450 <1> READ_DASD_TYPE:
4451 <1> READ_D_T:           ; GET DRIVE PARAMETERS
4452 <1>     PUSH    DS                 ; SAVE REGISTERS
4453 <1>     ;PUSH   ES
4454 <1>     PUSH    eBX
4455 <1>     ;CALL   DDS                 ; ESTABLISH ADDRESSING
4456 <1>     ;push   cs
4457 <1>     ;pop    ds
4458 <1>     mov     bx, KDATA
4459 <1>     mov     ds, bx
4460 <1>     ;mov    es, bx
4461 <1>     MOV     byte [DISK_STATUS1],0
4462 <1>     MOV     BL,[HF_NUM]           ; GET NUMBER OF DRIVES
4463 <1>     AND     DL,7FH                ; GET DRIVE NUMBER
4464 <1>     CMP     BL,DL
4465 <1>     JBE     short RDT_NOT_PRESENT   ; RETURN DRIVE NOT PRESENT
4466 <1>     CALL    GET_VEC                 ; GET DISK PARAMETER ADDRESS
4467 <1>     ;MOV    AL,[ES:BX+2]           ; HEADS
4468 <1>     mov     al, [ebx+2]
4469 <1>     ;MOV    CL,[ES:BX+14]
4470 <1>     mov     cl, [ebx+14]
4471 <1>     IMUL    CL                     ; * NUMBER OF SECTORS
4472 <1>     ;MOV    CX,[ES:BX]             ; MAX NUMBER OF CYLINDERS
4473 <1>     mov     cx ,[ebx]
4474 <1>     ;
4475 <1>     ; 02/01/2015
4476 <1>     ; ** leave the last cylinder as reserved for diagnostics **
4477 <1>     ; (Also in Award BIOS - 1999, AHDSK.ASM, FUN15 -> sub ax, 1)
4478 <1>     DEC     CX                     ; LEAVE ONE FOR DIAGNOSTICS
4479 <1>     ;
4480 <1>     IMUL    CX                     ; NUMBER OF SECTORS
4481 <1>     MOV     CX,DX                 ; HIGH ORDER HALF
4482 <1>     MOV     DX,AX                 ; LOW ORDER HALF
4483 <1>     ;SUB    AX,AX
4484 <1>     sub     al, al
4485 <1>     MOV     AH,03H                ; INDICATE FIXED DISK
4486 <1> RDT2: POP     eBX                 ; RESTORE REGISTERS
4487 <1>     ;POP    ES
4488 <1>     POP     DS
4489 <1>     ; (*) CLC                     ; CLEAR CARRY
4490 <1>     ;RETF   2
4491 <1>     ; (*) 29/05/2016
4492 <1>     ; (*) retf 4
4493 <1>     and     byte [esp+8], 0FEh ; clear carry bit of eflags register
4494 <1>     iretd
4495 <1>
4496 <1> RDT_NOT_PRESENT:
4497 <1>     SUB     AX,AX                 ; DRIVE NOT PRESENT RETURN
4498 <1>     MOV     CX,AX                 ; ZERO BLOCK COUNT
4499 <1>     MOV     DX,AX
4500 <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[C0380100] <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[BB380100]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[BC380100] <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 E8B3A00000 <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[BB380100]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[D0380100]      <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[4C5C0000]      <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[D0380100]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[BB380100]40 <1> CMP byte [DISK_STATUS1],BAD_SEEK
4729 000047D2 7507 <1> JNE short DS_EXIT
4730 000047D4 C605[BB380100]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[485C0000] <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[BB380100]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[BB380100]40 <1> CMP byte [DISK_STATUS1],BAD_SEEK ; SEEK NOT COMPLETE
4766 00004825 7507 <1> JNE short RECAL_EXIT ; IS OK
4767 00004827 C605[BB380100]00 <1> MOV byte [DISK_STATUS1],0
4768 <1> RECAL_EXIT:
4769 0000482E 803D[BB380100]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[485C0000] <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[485C0000] <1> mov dx, [HF_PORT]
```

```
4803 00004874 FEC2      <1>      inc     dl
4804 00004876 EC        <1>      IN      AL,DX
4805 00004877 A2[B2380100] <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_CNTLRL
4810                                <1> CD_EXIT:
4811 00004884 8825[BB380100] <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_ilx: ; 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[485C0000] <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[485C0000] <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_ilx ; 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[485C0000] <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[485C0000] <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[B1380100]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[485C0000] <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[BB380100]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[BB380100]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[BB380100]20 <1>      MOV     byte [DISK_STATUS1],BAD_CNTRLR
4951                      <1>  COMMAND4:
4952 00004975 5B           <1>      POP     eBX
4953 00004976 803D[BB380100]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[B3380100]00 <1>      MOV     byte [HF_INT_FLAG],0      ; RESET INTERRUPT FLAG
4959 00004987 FA           <1>      CLI
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[485C0000] <1>      mov     dx, [HF_PORT]
4975 000049A6 FEC2         <1>      inc     dl
4976 000049A8 F605[BD380100]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_CNTRLR
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[B3380100]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[BB380100]80 <1> WT2: MOV byte [DISK_STATUS1],TIME_OUT ; REPORT TIME OUT ERROR
5050 000049FA EB0E <1> JMP SHORT WT4
5051 000049FC C605[BB380100]00 <1> WT3: MOV byte [DISK_STATUS1],0
5052 00004A03 C605[B3380100]00 <1> MOV byte [HF_INT_FLAG],0
5053 00004A0A 803D[BB380100]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[485C0000] <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[BB380100] <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[485C0000] <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[BB380100]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[BB380100]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[485C0000] <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[B1380100] <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[BB380100] <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
5195      <1>
5196      <1> ;-----
5197      <1> ;      CHECK FIXED DISK ERROR REGISTER      :
5198      <1> ;-----
5199      <1> CHECK_ER:
5200      <1>      ;MOV     DX, HF_PORT+1      ; GET THE ERROR REGISTER
5201 00004AC3 668B15[485C0000] <1>      mov     dx, [HF_PORT]
5202 00004ACA FEC2      <1>      inc     dl
5203 00004ACC EC      <1>      IN      AL,DX

```

```

5204 00004ACD A2[B2380100]      <1>      MOV      [HF_ERROR],AL
5205 00004AD2 53                <1>      PUSH     eBX ; 21/02/2015
5206 00004AD3 B908000000        <1>      MOV      eCX,8          ; TEST ALL 8 BITS
5207 00004AD8 D0E0              <1> CK1:   SHL      AL,1          ; MOVE NEXT ERROR BIT TO CARRY
5208 00004ADA 7202              <1>      JC       short CK2      ; FOUND THE ERROR
5209 00004ADC E2FA              <1>      LOOP     CK1          ; KEEP TRYING
5210 00004ADE BB[3C5C0000]      <1> CK2:   MOV      eBX, ERR_TBL   ; COMPUTE ADDRESS OF
5211 00004AE3 01CB              <1>      ADD      eBX,eCX        ; ERROR CODE
5212                          <1>      ;;MOV    AH,BYTE [CS:BX]      ; GET ERROR CODE
5213                          <1>      ;mov     ah, [bx]
5214 00004AE5 8A23              <1>      mov     ah, [ebx] ; 21/02/2015
5215 00004AE7 8825[BB380100]    <1> CKEX:  MOV      [DISK_STATUS1],AH ; SAVE ERROR CODE
5216 00004AED 5B                <1>      POP      eBX
5217 00004AEE 80FC00            <1>      CMP      AH,0
5218 00004AF1 C3                <1>      RETn
5219                          <1>
5220                          <1> ;-----
5221                          <1> ; CHECK_DMA :
5222                          <1> ; -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :
5223                          <1> ; FIT WITHOUT SEGMENT OVERFLOW. :
5224                          <1> ; -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X :
5225                          <1> ; -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE) :
5226                          <1> ; -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) :
5227                          <1> ; -ERROR OTHERWISE :
5228                          <1> ;-----
5229                          <1> CHECK_DMA:
5230 00004AF2 6650              <1>      PUSH     AX          ; SAVE REGISTERS
5231 00004AF4 66B80080          <1>      MOV      AX,8000H      ; AH = MAX # SECTORS AL = MAX OFFSET
5232 00004AF8 F645FE02          <1>      TEST     byte [CMD_BLOCK+6],ECC_MODE
5233 00004AFC 7404              <1>      JZ       short CKD1
5234 00004AFE 66B8047F          <1>      MOV      AX,7F04H      ; ECC IS 4 MORE BYTES
5235 00004B02 3A65F9          <1> CKD1:  CMP      AH, [CMD_BLOCK+1] ; NUMBER OF SECTORS
5236 00004B05 7706              <1>      JA       short CKDOK    ; IT WILL FIT
5237 00004B07 7208              <1>      JB       short CKDERR    ; TOO MANY
5238 00004B09 38D8              <1>      CMP      AL,BL          ; CHECK OFFSET ON MAX SECTORS
5239 00004B0B 7204              <1>      JB       short CKDERR    ; ERROR
5240 00004B0D F8                <1> CKDOK:  CLC              ; CLEAR CARRY
5241 00004B0E 6658              <1>      POP      AX
5242 00004B10 C3                <1>      RETn          ; NORMAL RETURN
5243 00004B11 F9                <1> CKDERR: STC              ; INDICATE ERROR
5244 00004B12 C605[BB380100]09   <1>      MOV      byte [DISK_STATUS1],DMA_BOUNDARY
5245 00004B19 6658              <1>      POP      AX
5246 00004B1B C3                <1>      RETn
5247                          <1>
5248                          <1> ;-----
5249                          <1> ; SET UP ES:BX-> DISK PARMS :
5250                          <1> ;-----
5251                          <1>
5252                          <1> ; INPUT -> DL = 0 based drive number
5253                          <1> ; OUTPUT -> ES:BX = disk parameter table address
5254                          <1>
5255                          <1> GET_VEC:
5256                          <1> ;SUB     AX,AX          ; GET DISK PARAMETER ADDRESS
5257                          <1> ;MOV      ES,AX
5258                          <1> ;TEST     DL,1
5259                          <1> ;JZ       short GV_0
5260                          <1> ; LES      BX,[HF1_TBL_VEC] ; ES:BX -> DRIVE PARAMETERS
5261                          <1> ; JMP      SHORT GV_EXIT
5262                          <1> ;GV_0:
5263                          <1> ; LES      BX,[HF_TBL_VEC] ; ES:BX -> DRIVE PARAMETERS
5264                          <1> ;
5265                          <1> ;xor      bh, bh
5266 00004B1C 31DB              <1>      xor      ebx, ebx
5267 00004B1E 88D3              <1>      mov      bl, dl
5268                          <1> ;;02/01/2015
5269                          <1> ;;shl     bl, 1          ; port address offset
5270                          <1> ;;mov     ax, [bx+hd_ports] ; Base port address (1F0h, 170h)
5271                          <1> ;;shl     bl, 1          ; dpt pointer offset
5272 00004B20 C0E302            <1>      shl     bl, 2 ;;
5273                          <1> ;add     bx, HF_TBL_VEC ; Disk parameter table pointer
5274 00004B23 81C3[C0380100]    <1>      add     ebx, HF_TBL_VEC ; 21/02/2015
5275                          <1> ;push     word [bx+2] ; dpt segment
5276                          <1> ;pop      es
5277                          <1> ;mov      bx, [bx] ; dpt offset
5278 00004B29 8B1B              <1>      mov     ebx, [ebx]
5279                          <1> ;GV_EXIT:
5280 00004B2B C3                <1>      RETn
5281                          <1>
5282                          <1> hdc1_int: ; 21/02/2015
5283                          <1> ;--- HARDWARE INT 76H -- ( IRQ LEVEL 14 ) -----
5284                          <1> ; :
5285                          <1> ; FIXED DISK INTERRUPT ROUTINE :
5286                          <1> ; :
5287                          <1> ;-----
5288                          <1>
5289                          <1> ; 22/12/2014
5290                          <1> ; IBM PC-XT Model 286 System BIOS Source Code - DISK.ASM (HD_INT)
5291                          <1> ; '11/15/85'
5292                          <1> ; AWARD BIOS 1999 (D1A0622)
5293                          <1> ; Source Code - ATORGS.ASM (INT_HDISK, INT_HDISK1)
5294                          <1>
5295                          <1> ;int_76h:
5296                          <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[B3380100]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>      ; (Award BIOS - 1999)
5312 00004B42 EC <1>      Clear_IRQ1415:
5313 00004B43 665A <1>      in      al, dx ;
5314 <1>      pop     dx
5314 00004B45 E6EB <1>      NEWIODELAY
5315 <2>      out 0ebh,al
5316 00004B47 B020 <1>      ;
5317 00004B49 E6A0 <1>      MOV     AL,EOI ; NON-SPECIFIC END OF INTERRUPT
5318 <1>      OUT     INTB00,AL ; FOR CONTROLLER #2
5319 <1>      ;JMP     $+2 ; WAIT
5319 00004B4B E6EB <1>      NEWIODELAY
5320 00004B4D E620 <2>      out 0ebh,al
5321 00004B4F 1F <1>      OUT     INTA00,AL ; FOR CONTROLLER #1
5322 <1>      POP     DS
5323 <1>      ;STI ; RE-ENABLE INTERRUPTS
5324 <1>      ;MOV     AX,9100H ; DEVICE POST
5325 <1>      ;INT     15H ; INTERRUPT
5326 00004B50 6658 <1>      irq15_iret: ; 25/02/2015
5327 00004B52 CF <1>      POP     AX
5328 <1>      IRETD ; RETURN FROM INTERRUPT
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[B3380100]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: 26/03/2017
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> ;
58 <1> ;
59 <1> ;
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 response 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 response 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):
230 <1> ;; # Parent's read only pages are copied to new child pages.
231 <1> ;; Parent's PTE attributes are not changed.
232 <1> ;; (Because, there is another parent-child fork before this fork! We must not
233 <1> ;; destroy/mix previous fork result).
234 <1> ;; # Child's Page Table Entries (which are corresponding to Parent's
235 <1> ;; read only pages) are set as writable (while duplicated PTE bit is clear).
236 <1> ;; # Parent's PTEs with writable page attribute are updated to point same pages
237 <1> ;; as read only, (while) duplicated PTE bit is reset (clear).
238 <1> ;; # Parent's Page Table Entries (with writable page attribute) are duplicated

```

```

239 <1> ;; as Child's Page Table Entries without copying actual page.
240 <1> ;; # Child 's Page Table Entries (which are corresponding to Parent's writable
241 <1> ;; pages) are updated as duplicated PTE bit (AVL bit 0, PTE bit 9- is set.
242 <1> ;;
243 <1> ;; !? WHAT FOR (duplication after duplication):
244 <1> ;; In UNIX method for sys fork (a typical 'fork' application in /etc/init)
245 <1> ;; program/executable code continues from specified location as child process,
246 <1> ;; returns back previous code location as parent process, every child after
247 <1> ;; every sys fork uses last image of code and data just prior the fork.
248 <1> ;; Even if the parent code changes data, the child will not see the changed data
249 <1> ;; after the fork. In Retro UNIX 8086 v1, parent's process segment (32KB)
250 <1> ;; was copied to child's process segment (all of code and data) according to
251 <1> ;; original UNIX v1 which copies all of parent process code and data -core-
252 <1> ;; to child space -core- but swaps that core image -of child- on to disk.
253 <1> ;; If I (Erdogan Tan) would use a method of to copy parent's core
254 <1> ;; (complete running image of parent process) to the child process;
255 <1> ;; for big sizes, i would force Retro UNIX 386 v1 to spend many memory pages
256 <1> ;; and times only for a sys fork. (It would excessive reservation for sys fork,
257 <1> ;; because sys fork usually is prior to sys exec; sys exec always establishes
258 <1> ;; a new/fresh core -running space-, by clearing all code/data content).
259 <1> ;; 'Read Only' page flag ensures page fault handler is needed only for a few write
260 <1> ;; attempts between sys fork and sys exec, not more... (I say so by thinking
261 <1> ;; of "/etc/init" content, specially.) sys exec will clear page tables and
262 <1> ;; new/fresh pages will be used to load and run new executable/program.
263 <1> ;; That is what for i have preferred "copy on write", "duplication" method
264 <1> ;; for sharing same read only pages between parent and child processes.
265 <1> ;; That is a pity i have to use new private flag (AVL bit 0, "duplicated PTE
266 <1> ;; belongs to child" sign) for cooperation on duplicated pages between a parent
267 <1> ;; and it's child processes; otherwise parent process would destroy data belongs
268 <1> ;; to its child or vice versa; or some pages would remain unclaimed
269 <1> ;; -deallocation problem-.
270 <1> ;; Note: to prevent conflicts, read only pages must not be swapped out...
271 <1> ;;
272 <1> ;; WHEN PARENT TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
273 <1> ;; # Page fault handler will do those:
274 <1> ;; - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
275 <1> ;; - If it is reset/clear, there is a child uses same page.
276 <1> ;; - Parent's read only page -previous page- is copied to a new writable page.
277 <1> ;; - Parent's PTE is updated as writable page, as unique page (AVL=0)
278 <1> ;; - (Page fault handler whill check this PTE later, if child process causes to
279 <1> ;; page fault due to write attempt on read only page. Of course, the previous
280 <1> ;; read only page will be converted to writable and unique page which belongs
281 <1> ;; to child process.)
282 <1> ;; WHEN CHILD TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
283 <1> ;; # Page fault handler will do those:
284 <1> ;; - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
285 <1> ;; - If it is set, there is a parent uses -or was using- same page.
286 <1> ;; - Same PTE address within parent's page table is checked if it has same page
287 <1> ;; address or not.
288 <1> ;; - If parent's PTE has same address, child will continue with a new writable page.
289 <1> ;; Parent's PTE will point to same (previous) page as writable, unique (AVL=0).
290 <1> ;; - If parent's PTE has different address, child will continue with it's
291 <1> ;; own/same page but read only flag (0) will be changed to writable flag (1) and
292 <1> ;; 'duplicated PTE (belongs to child)' flag/sign will be cleared/reset.
293 <1> ;;
294 <1> ;; NOTE: When a child process is terminated, read only flags of parent's page tables
295 <1> ;; will be set as writable (and unique) in case of child process was using
296 <1> ;; same pages with duplicated child PTE sign... Depending on sys fork and
297 <1> ;; duplication method details, it is not possible multiple child processes
298 <1> ;; were using same page with duplicated PTEs.
299 <1> ;;
300 <1> ;*****
301 <1>
302 <1> ;; 08/10/2014
303 <1> ;; 11/09/2014 - Retro UNIX 386 v1 PAGING (further) draft
304 <1> ;; by Erdogan Tan (Based on KolibriOS 'memory.inc')
305 <1>
306 <1> ;; 'allocate_page' code is derived and modified from KolibriOS
307 <1> ;; 'alloc_page' procedure in 'memory.inc'
308 <1> ;; (25/08/2014, Revision: 5057) file
309 <1> ;; by KolibriOS Team (2004-2012)
310 <1>
311 <1> allocate_page:
312 <1> ; 01/07/2015
313 <1> ; 05/05/2015
314 <1> ; 30/04/2015
315 <1> ; 16/10/2014
316 <1> ; 08/10/2014
317 <1> ; 09/09/2014 (Retro UNIX 386 v1 - beginning)
318 <1> ;
319 <1> ; INPUT -> none
320 <1> ;
321 <1> ; OUTPUT ->
322 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
323 <1> ; (corresponding MEMORY ALLOCATION TABLE bit is RESET)
324 <1> ;
325 <1> ; CF = 1 and EAX = 0
326 <1> ; if there is not a free page to be allocated
327 <1> ;
328 <1> ; Modified Registers -> none (except EAX)
329 <1> ;
330 00004B79 A1[28380100] <1> mov eax, [free_pages]
331 00004B7E 21C0 <1> and eax, eax
332 00004B80 7438 <1> jz short out_of_memory
333 <1> ;
334 00004B82 53 <1> push ebx
335 00004B83 51 <1> push ecx
336 <1> ;
337 00004B84 BB00001000 <1> mov ebx, MEM_ALLOC_TBL ; Memory Allocation Table offset
338 00004B89 89D9 <1> mov ecx, ebx
339 <1> ; NOTE: 32 (first_page) is initial
340 <1> ; value of [next_page].

```

```

341                                     <1>                                     ; It points to the first available
342                                     <1>                                     ; page block for users (ring 3) ...
343                                     <1>                                     ; (MAT offset 32 = 1024/32)
344                                     <1>                                     ; (at the of the first 4 MB)
345 00004B8B 031D[2C380100]          <1>         add     ebx, [next_page] ; Free page searching starts from here
346                                     <1>                                     ; next_free_page >> 5
347 00004B91 030D[30380100]          <1>         add     ecx, [last_page] ; Free page searching ends here
348                                     <1>                                     ; (total_pages - 1) >> 5
349                                     <1> al_p_scan:
350 00004B97 39CB                    <1>         cmp     ebx, ecx
351 00004B99 770A                    <1>         ja     short al_p_notfound
352                                     <1>                                     ;
353                                     <1>                                     ; 01/07/2015
354                                     <1>                                     ; AMD64 Architecture Programmer's Manual
355                                     <1>                                     ; Volume 3:
356                                     <1>                                     ; General-Purpose and System Instructions
357                                     <1>                                     ;
358                                     <1>                                     ; BSF - Bit Scan Forward
359                                     <1>                                     ;
360                                     <1>                                     ; Searches the value in a register or a memory location
361                                     <1>                                     ; (second operand) for the least-significant set bit.
362                                     <1>                                     ; If a set bit is found, the instruction clears the zero flag (ZF)
363                                     <1>                                     ; and stores the index of the least-significant set bit in a destination
364                                     <1>                                     ; register (first operand). If the second operand contains 0,
365                                     <1>                                     ; the instruction sets ZF to 1 and does not change the contents of the
366                                     <1>                                     ; destination register. The bit index is an unsigned offset from bit 0
367                                     <1>                                     ; of the searched value
368                                     <1>                                     ;
369 00004B9B 0FBC03                  <1>         bsf     eax, [ebx] ; Scans source operand for first bit set (1).
370                                     <1>                                     ; Clear ZF if a bit is found set (1) and
371                                     <1>                                     ; loads the destination with an index to
372                                     <1>                                     ; first set bit. (0 -> 31)
373                                     <1>                                     ; Sets ZF to 1 if no bits are found set.
374 00004B9E 7525                  <1>         jnz     short al_p_found ; ZF = 0 -> a free page has been found
375                                     <1>                                     ;
376                                     <1>                                     ; NOTE: a Memory Allocation Table bit
377                                     <1>                                     ; with value of 1 means
378                                     <1>                                     ; the corresponding page is free
379                                     <1>                                     ; (Retro UNIX 386 v1 feature only!)
380 00004BA0 83C304                  <1>         add     ebx, 4
381                                     <1>                                     ; We return back for searching next page block
382                                     <1>                                     ; NOTE: [free_pages] is not ZERO; so,
383                                     <1>                                     ; we always will find at least 1 free page here.
384 00004BA3 EBF2                    <1>         jmp     short al_p_scan
385                                     <1>                                     ;
386                                     <1> al_p_notfound:
387 00004BA5 81E900001000            <1>         sub     ecx, MEM_ALLOC_TBL
388 00004BAB 890D[2C380100]          <1>         mov     [next_page], ecx ; next/first free page = last page
389                                     <1>                                     ; (deallocate_page procedure will change it)
390 00004BB1 31C0                    <1>         xor     eax, eax
391 00004BB3 A3[28380100]            <1>         mov     [free_pages], eax ; 0
392 00004BB8 59                      <1>         pop     ecx
393 00004BB9 5B                      <1>         pop     ebx
394                                     <1>                                     ;
395                                     <1> out_of_memory:
396 00004BBA E85B040000            <1>         call    swap_out
397 00004BBF 7325                    <1>         jnc     short al_p_ok ; [free_pages] = 0, re-allocation by swap_out
398                                     <1>                                     ;
399 00004BC1 29C0                    <1>         sub     eax, eax ; 0
400 00004BC3 F9                      <1>         stc
401 00004BC4 C3                      <1>         retn
402                                     <1>                                     ;
403                                     <1> al_p_found:
404 00004BC5 89D9                    <1>         mov     ecx, ebx
405 00004BC7 81E900001000            <1>         sub     ecx, MEM_ALLOC_TBL
406 00004BCD 890D[2C380100]          <1>         mov     [next_page], ecx ; Set first free page searching start
407                                     <1>                                     ; address/offset (to the next)
408 00004BD3 FF0D[28380100]          <1>         dec     dword [free_pages] ; 1 page has been allocated (X = X-1)
409                                     <1>                                     ;
410 00004BD9 0FB303                  <1>         btr     [ebx], eax ; The destination bit indexed by the source value
411                                     <1>                                     ; is copied into the Carry Flag and then cleared
412                                     <1>                                     ; in the destination.
413                                     <1>                                     ;
414                                     <1>                                     ; Reset the bit which is corresponding to the
415                                     <1>                                     ; (just) allocated page.
416                                     <1>                                     ; 01/07/2015 (4*8 = 32, 1 allocation byte = 8 pages)
417 00004BDC C1E103                  <1>         shl     ecx, 3 ; (page block offset * 32) + page index
418 00004BDF 01C8                    <1>         add     eax, ecx ; = page number
419 00004BE1 C1E00C                  <1>         shl     eax, 12 ; physical address of the page (flat/real value)
420                                     <1>                                     ; EAX = physical address of memory page
421                                     <1>                                     ;
422                                     <1>                                     ; NOTE: The relevant page directory and page table entry will be updated
423                                     <1>                                     ; according to this EAX value...
424 00004BE4 59                      <1>         pop     ecx
425 00004BE5 5B                      <1>         pop     ebx
426                                     <1> al_p_ok:
427 00004BE6 C3                      <1>         retn
428                                     <1>                                     ;
429                                     <1>                                     ;
430                                     <1> make_page_dir:
431                                     <1>                                     ; 18/04/2015
432                                     <1>                                     ; 12/04/2015
433                                     <1>                                     ; 23/10/2014
434                                     <1>                                     ; 16/10/2014
435                                     <1>                                     ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
436                                     <1>                                     ;
437                                     <1>                                     ; INPUT ->
438                                     <1>                                     ; none
439                                     <1>                                     ; OUTPUT ->
440                                     <1>                                     ; (EAX = 0)
441                                     <1>                                     ; cf = 1 -> insufficient (out of) memory error
442                                     <1>                                     ; cf = 0 ->

```

```

443      <1>      ;      u.pgdir = page directory (physical) address of the current
444      <1>      ;      process/user.
445      <1>      ;
446      <1>      ; Modified Registers -> EAX
447      <1>      ;
448 00004BE7 E88DFFFFFF      <1>      call    allocate_page
449 00004BEC 7216           <1>      jc      short mkpd_error
450      <1>      ;
451 00004BEE A3[004F0100]    <1>      mov     [u.pgdir], eax      ; Page dir address for current user/process
452      <1>      ; (Physical address)
453      <1> clear_page:
454      <1>      ; 18/04/2015
455      <1>      ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
456      <1>      ;
457      <1>      ; INPUT ->
458      <1>      ;      EAX = physical address of the page
459      <1>      ; OUTPUT ->
460      <1>      ;      all bytes of the page will be cleared
461      <1>      ;
462      <1>      ; Modified Registers -> none
463      <1>      ;
464 00004BF3 57             <1>      push    edi
465 00004BF4 51             <1>      push    ecx
466 00004BF5 50             <1>      push    eax
467 00004BF6 B900040000     <1>      mov     ecx, PAGE_SIZE / 4
468 00004BFB 89C7           <1>      mov     edi, eax
469 00004BFD 31C0           <1>      xor     eax, eax
470 00004BFF F3AB           <1>      rep     stosd
471 00004C01 58             <1>      pop     eax
472 00004C02 59             <1>      pop     ecx
473 00004C03 5F             <1>      pop     edi
474      <1> mkpd_error:
475      <1> mkpt_error:
476 00004C04 C3             <1>      retn
477      <1>
478      <1> make_page_table:
479      <1>      ; 23/06/2015
480      <1>      ; 18/04/2015
481      <1>      ; 12/04/2015
482      <1>      ; 16/10/2014
483      <1>      ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
484      <1>      ;
485      <1>      ; INPUT ->
486      <1>      ;      EBX = virtual (linear) address
487      <1>      ;      ECX = page table attributes (lower 12 bits)
488      <1>      ;      (higher 20 bits must be ZERO)
489      <1>      ;      (bit 0 must be 1)
490      <1>      ;      u.pgdir = page directory (physical) address
491      <1>      ; OUTPUT ->
492      <1>      ;      EDX = Page directory entry address
493      <1>      ;      EAX = Page table address
494      <1>      ;      cf = 1 -> insufficient (out of) memory error
495      <1>      ;      cf = 0 -> page table address in the PDE (EDX)
496      <1>      ;
497      <1>      ; Modified Registers -> EAX, EDX
498      <1>      ;
499 00004C05 E86FFFFFFF     <1>      call    allocate_page
500 00004C0A 72F8           <1>      jc      short mkpt_error
501 00004C0C E811000000     <1>      call    set_pde
502 00004C11 EBEO           <1>      jmp     short clear_page
503      <1>
504      <1> make_page:
505      <1>      ; 24/07/2015
506      <1>      ; 23/06/2015 ; (Retro UNIX 386 v1 - beginning)
507      <1>      ;
508      <1>      ; INPUT ->
509      <1>      ;      EBX = virtual (linear) address
510      <1>      ;      ECX = page attributes (lower 12 bits)
511      <1>      ;      (higher 20 bits must be ZERO)
512      <1>      ;      (bit 0 must be 1)
513      <1>      ;      u.pgdir = page directory (physical) address
514      <1>      ; OUTPUT ->
515      <1>      ;      EBX = Virtual address
516      <1>      ;      (EDX = PTE value)
517      <1>      ;      EAX = Physical address
518      <1>      ;      cf = 1 -> insufficient (out of) memory error
519      <1>      ;
520      <1>      ; Modified Registers -> EAX, EDX
521      <1>      ;
522 00004C13 E861FFFFFF     <1>      call    allocate_page
523 00004C18 7207           <1>      jc      short mkp_err
524 00004C1A E821000000     <1>      call    set_pte
525 00004C1F 73D2           <1>      jnc     short clear_page ; 18/04/2015
526      <1> mkp_err:
527 00004C21 C3             <1>      retn
528      <1>
529      <1>
530      <1> set_pde:      ; Set page directory entry (PDE)
531      <1>      ; 20/07/2015
532      <1>      ; 18/04/2015
533      <1>      ; 12/04/2015
534      <1>      ; 23/10/2014
535      <1>      ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
536      <1>      ;
537      <1>      ; INPUT ->
538      <1>      ;      EAX = physical address
539      <1>      ;      (use present value if EAX = 0)
540      <1>      ;      EBX = virtual (linear) address
541      <1>      ;      ECX = page table attributes (lower 12 bits)
542      <1>      ;      (higher 20 bits must be ZERO)
543      <1>      ;      (bit 0 must be 1)
544      <1>      ;      u.pgdir = page directory (physical) address

```



```

545      <1>      ; OUTPUT ->
546      <1>      ;      EDX = PDE address
547      <1>      ;      EAX = page table address (physical)
548      <1>      ;      ; (CF=1 -> Invalid page address)
549      <1>      ;
550      <1>      ; Modified Registers -> EDX
551      <1>      ;
552      00004C22 89DA      <1>      mov     edx, ebx
553      00004C24 C1EA16    <1>      shr     edx, PAGE_D_SHIFT ; 22
554      00004C27 C1E202    <1>      shl     edx, 2 ; offset to page directory (1024*4)
555      00004C2A 0315[004F0100] <1>      add     edx, [u.pgdir]
556      <1>      ;
557      00004C30 21C0      <1>      and     eax, eax
558      00004C32 7506      <1>      jnz     short spde_1
559      <1>      ;
560      00004C34 8B02      <1>      mov     eax, [edx] ; old PDE value
561      <1>      ; test al, 1
562      <1>      ; jz     short spde_2
563      00004C36 662500F0  <1>      and     ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
564      <1>      spde_1:
565      <1>      ; and     cx, 0FFFh
566      00004C3A 8902      <1>      mov     [edx], eax
567      00004C3C 66090A    <1>      or      [edx], cx
568      00004C3F C3       <1>      retn
569      <1>      ; spde_2: ; error
570      <1>      ; stc
571      <1>      ; retn
572      <1>
573      <1>      set_pte:      ; Set page table entry (PTE)
574      <1>      ; 24/07/2015
575      <1>      ; 20/07/2015
576      <1>      ; 23/06/2015
577      <1>      ; 18/04/2015
578      <1>      ; 12/04/2015
579      <1>      ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
580      <1>      ;
581      <1>      ; INPUT ->
582      <1>      ;      EAX = physical page address
583      <1>      ;      (use present value if EAX = 0)
584      <1>      ;      EBX = virtual (linear) address
585      <1>      ;      ECX = page attributes (lower 12 bits)
586      <1>      ;      (higher 20 bits must be ZERO)
587      <1>      ;      (bit 0 must be 1)
588      <1>      ;      u.pgdir = page directory (physical) address
589      <1>      ; OUTPUT ->
590      <1>      ;      EAX = physical page address
591      <1>      ;      (EDX = PTE value)
592      <1>      ;      EBX = virtual address
593      <1>      ;
594      <1>      ;      CF = 1 -> error
595      <1>      ;
596      <1>      ; Modified Registers -> EAX, EDX
597      <1>      ;
598      00004C40 50       <1>      push    eax
599      00004C41 A1[004F0100] <1>      mov     eax, [u.pgdir] ; 20/07/2015
600      00004C46 E837000000 <1>      call    get_pde
601      <1>      ; EDX = PDE address
602      <1>      ; EAX = PDE value
603      00004C4B 5A       <1>      pop     edx ; physical page address
604      00004C4C 722A      <1>      jc      short spte_err ; PDE not present
605      <1>      ;
606      00004C4E 53       <1>      push    ebx ; 24/07/2015
607      00004C4F 662500F0  <1>      and     ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
608      <1>      ; EDX = PT address (physical)
609      00004C53 C1EB0C    <1>      shr     ebx, PAGE_SHIFT ; 12
610      00004C56 81E3FF030000 <1>      and     ebx, PTE_MASK ; 03FFh
611      <1>      ; clear higher 10 bits (PD bits)
612      00004C5C C1E302    <1>      shl     ebx, 2 ; offset to page table (1024*4)
613      00004C5F 01C3      <1>      add     ebx, eax
614      <1>      ;
615      00004C61 8B03      <1>      mov     eax, [ebx] ; Old PTE value
616      00004C63 A801      <1>      test    al, 1
617      00004C65 740C      <1>      jz      short spte_0
618      00004C67 09D2      <1>      or      edx, edx
619      00004C69 750F      <1>      jnz     short spte_1
620      00004C6B 662500F0  <1>      and     ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 bits
621      00004C6F 89C2      <1>      mov     edx, eax
622      00004C71 EB09      <1>      jmp     short spte_2
623      <1>      spte_0:
624      <1>      ; If this PTE contains a swap (disk) address,
625      <1>      ; it can be updated by using 'swap_in' procedure
626      <1>      ; only!
627      00004C73 21C0      <1>      and     eax, eax
628      00004C75 7403      <1>      jz      short spte_1
629      <1>      ; 24/07/2015
630      <1>      ; swapped page ! (on disk)
631      00004C77 5B       <1>      pop     ebx
632      <1>      spte_err:
633      00004C78 F9       <1>      stc
634      00004C79 C3       <1>      retn
635      <1>      spte_1:
636      00004C7A 89D0      <1>      mov     eax, edx
637      <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 gpde_retn
667      00004C92 F9        <1>      stc
668      <1>      gpde_retn:
669      00004C93 C3        <1>      retn
670
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 gpde_1
693      <1>      ;retn
694      <1>      ;gpde_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>      gpde_retn:
704      00004CB1 C3        <1>      retn
705
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
740      <1>      dapd_1:
741      00004CC9 41        <1>      inc     ecx ; page directory entry index
742      00004CCA 81F900040000 <1>      cmp     ecx, PAGE_SIZE / 4 ; 1024
743      00004CD0 72E9      <1>      jb      short dapd_0
744      <1>      dapd_2:
745      00004CD2 58        <1>      pop     eax
746      00004CD3 E87F000000    <1>      call    deallocate_page ; deallocate the page dir's itself
747      00004CD8 59        <1>      pop     ecx
748      00004CD9 5E        <1>      pop     esi

```

```

749 00004CDA C3      <1>      retn
750                  <1>
751                  <1> deallocate_page_table:
752                  <1>      ; 12/07/2016
753                  <1>      ; 19/09/2015
754                  <1>      ; 15/09/2015
755                  <1>      ; 05/08/2015
756                  <1>      ; 30/04/2015
757                  <1>      ; 28/04/2015
758                  <1>      ; 24/10/2014
759                  <1>      ; 23/10/2014
760                  <1>      ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
761                  <1>      ;
762                  <1>      ; INPUT ->
763                  <1>      ;      EAX = PHYSICAL (real/flat) ADDRESS OF THE PAGE TABLE
764                  <1>      ;      EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
765                  <1>      ;      (ECX = page directory entry index)
766                  <1>      ; OUTPUT ->
767                  <1>      ;      All of pages in the page table and page table's itself
768                  <1>      ;      will be deallocated except 'read only' duplicated pages
769                  <1>      ;      (will be converted to writable pages).
770                  <1>      ;
771                  <1>      ; Modified Registers -> EAX
772                  <1>      ;
773 00004CDB 56      <1>      push     esi
774 00004CDC 57      <1>      push     edi
775 00004CDD 52      <1>      push     edx
776 00004CDE 50      <1>      push     eax ; *
777 00004CDF 89C6    <1>      mov      esi, eax
778 00004CE1 31FF    <1>      xor      edi, edi ; 0
779                  <1> dapt_0:
780 00004CE3 AD      <1>      lodsd
781 00004CE4 A801    <1>      test     al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
782 00004CE6 7441    <1>      jz       short dapt_1
783                  <1>      ;
784 00004CE8 A802    <1>      test     al, PTE_A_WRITE   ; bit 1, writable (r/w) flag
785                  <1>      ; (must be 1)
786 00004CEA 754C    <1>      jnz      short dapt_3
787                  <1>      ; Read only -duplicated- page (belongs to a parent or a child)
788 00004CEC 66A90002 <1>      test     ax, PTE_DUPLICATED ; Was this page duplicated
789                  <1>      ; as child's page ?
790 00004CF0 7451    <1>      jz       short dapt_4 ; Clear PTE but don't deallocate the page!
791                  <1>      ; check the parent's PTE value is read only & same page or not..
792                  <1>      ; ECX = page directory entry index (0-1023)
793 00004CF2 53      <1>      push     ebx
794 00004CF3 51      <1>      push     ecx
795 00004CF4 66C1E102 <1>      shl      cx, 2 ; *4
796 00004CF8 01CB    <1>      add      ebx, ecx ; PDE offset (for the parent)
797 00004CFA 8B0B    <1>      mov      ecx, [ebx]
798 00004CFC F6C101  <1>      test     cl, PDE_A_PRESENT ; present (valid) or not ?
799 00004CFF 7435    <1>      jz       short dapt_2 ; parent process does not use this page
800 00004D01 6681E100F0 <1>      and      cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
801                  <1>      ; EDI = page table entry index (0-1023)
802 00004D06 89FA    <1>      mov      edx, edi
803 00004D08 66C1E202 <1>      shl      dx, 2 ; *4
804 00004D0C 01CA    <1>      add      edx, ecx ; PTE offset (for the parent)
805 00004D0E 8B1A    <1>      mov      ebx, [edx]
806 00004D10 F6C301  <1>      test     bl, PTE_A_PRESENT ; present or not ?
807 00004D13 7421    <1>      jz       short dapt_2 ; parent process does not use this page
808 00004D15 662500F0 <1>      and      ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
809 00004D19 6681E300F0 <1>      and      bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
810 00004D1E 39D8    <1>      cmp      eax, ebx ; parent's and child's pages are same ?
811 00004D20 7514    <1>      jne      short dapt_2 ; not same page
812                  <1>      ; deallocate the child's page
813 00004D22 800A02  <1>      or       byte [edx], PTE_A_WRITE ; convert to writable page (parent)
814 00004D25 59      <1>      pop      ecx
815 00004D26 5B      <1>      pop      ebx
816 00004D27 EB1A    <1>      jmp      short dapt_4
817                  <1> dapt_1:
818 00004D29 09C0    <1>      or       eax, eax ; swapped page ?
819 00004D2B 741D    <1>      jz       short dapt_5 ; no
820                  <1>      ; yes
821 00004D2D D1E8    <1>      shr      eax, 1
822 00004D2F E8CA040000 <1>      call     unlink_swap_block ; Deallocate swapped page block
823                  <1>      ; on the swap disk (or in file)
824 00004D34 EB14    <1>      jmp      short dapt_5
825                  <1> dapt_2:
826 00004D36 59      <1>      pop      ecx
827 00004D37 5B      <1>      pop      ebx
828                  <1> dapt_3:
829                  <1>      ; 12/07/2016
830 00004D38 66A90004 <1>      test     ax, PTE_SHARED ; shared or direct memory access indicator
831 00004D3C 7505    <1>      jnz      short dapt_4 ; AVL bit 1 = 1, do not deallocate this page!
832                  <1>      ;
833                  <1>      ;and ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
834 00004D3E E814000000 <1>      call     deallocate_page ; set the mem allocation bit of this page
835                  <1> dapt_4:
836 00004D43 C746FC00000000 <1>      mov      dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
837                  <1> dapt_5:
838 00004D4A 47      <1>      inc      edi ; page table entry index
839 00004D4B 81FF00040000 <1>      cmp      edi, PAGE_SIZE / 4 ; 1024
840 00004D51 7290    <1>      jb       short dapt_0
841                  <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[2C380100] <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[2C380100] <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>      ;      ; complement carry flag
895      <1>      ;      ; do not increase free_pages count
896      <1>      ;      ; if the page is already deallocated
897      <1>      ;      ; before.
898      00004D7F FF05[28380100] <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>      ;;.ll:
928      <1>      ;;      bsf     eax, [ebx];
929      <1>      ;;      jnz     .found
930      <1>      ;;      add     ebx, 4
931      <1>      ;;      cmp     ebx, ecx
932      <1>      ;;      jb     .ll
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
944      <1>      ;;      sub     ebx, sys_pgmap
945      <1>      ;;      lea     eax, [eax+ebx*8]
946      <1>      ;;      shl     eax, 12
947      <1>      ;;;//-      dec [pg_data.pages_free]
948      <1>      ;;      pop     ebx
949      <1>      ;;      popfd
950      <1>      ;;      ret
951      <1>      ;;;//-
952      <1>      ;;.out_of_memory:

```



```

953 <1> ;;      mov      [pg_data.pages_free], 1
954 <1> ;;      xor      eax, eax
955 <1> ;;      pop      ebx
956 <1> ;;      popfd
957 <1> ;;      ret
958 <1> ;;;// -
959 <1> ;;endp
960 <1>
961 <1> duplicate_page_dir:
962 <1>      ; 21/09/2015
963 <1>      ; 31/08/2015
964 <1>      ; 20/07/2015
965 <1>      ; 28/04/2015
966 <1>      ; 27/04/2015
967 <1>      ; 18/04/2015
968 <1>      ; 12/04/2015
969 <1>      ; 18/10/2014
970 <1>      ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
971 <1>      ;
972 <1>      ; INPUT ->
973 <1>      ;      [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
974 <1>      ;      page directory.
975 <1>      ; OUTPUT ->
976 <1>      ;      EAX = PHYSICAL (real/flat) ADDRESS of the child's
977 <1>      ;      page directory.
978 <1>      ;      (New page directory with new page table entries.)
979 <1>      ;      (New page tables with read only copies of the parent's
980 <1>      ;      pages.)
981 <1>      ;      EAX = 0 -> Error (CF = 1)
982 <1>      ;
983 <1>      ; Modified Registers -> none (except EAX)
984 <1>      ;
985 00004D88 E8ECFDFFFF <1>      call  allocate_page
986 00004D8D 723E <1>      jc     short dpd_err
987 <1>      ;
988 00004D8F 55 <1>      push  ebp ; 20/07/2015
989 00004D90 56 <1>      push  esi
990 00004D91 57 <1>      push  edi
991 00004D92 53 <1>      push  ebx
992 00004D93 51 <1>      push  ecx
993 00004D94 8B35[004F0100] <1>      mov   esi, [u.pgdir]
994 00004D9A 89C7 <1>      mov   edi, eax
995 00004D9C 50 <1>      push  eax ; save child's page directory address
996 <1>      ; 31/08/2015
997 <1>      ; copy PDE 0 from the parent's page dir to the child's page dir
998 <1>      ; (use same system space for all user page tables)
999 00004D9D A5 <1>      movsd
1000 00004D9E BD00004000 <1>      mov   ebp, 1024*4096 ; pass the 1st 4MB (system space)
1001 00004DA3 B9FF030000 <1>      mov   ecx, (PAGE_SIZE / 4) - 1 ; 1023
1002 <1> dpd_0:
1003 00004DA8 AD <1>      lodsd
1004 <1>      ;or      eax, eax
1005 <1>      ;jnz     short dpd_1
1006 00004DA9 A801 <1>      test  al, PDE_A_PRESENT ; bit 0 = 1
1007 00004DAB 7508 <1>      jnz   short dpd_1
1008 <1>      ; 20/07/2015 (virtual address at the end of the page table)
1009 00004DAD 81C500004000 <1>      add   ebp, 1024*4096 ; page size * PTE count
1010 00004DB3 EB0F <1>      jmp   short dpd_2
1011 <1> dpd_1:
1012 00004DB5 662500F0 <1>      and   ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
1013 00004DB9 89C3 <1>      mov   ebx, eax
1014 <1>      ; EBX = Parent's page table address
1015 00004DBB E81F000000 <1>      call  duplicate_page_table
1016 00004DC0 720C <1>      jc     short dpd_p_err
1017 <1>      ; EAX = Child's page table address
1018 00004DC2 0C07 <1>      or    al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
1019 <1>      ; set bit 0, bit 1 and bit 2 to 1
1020 <1>      ; (present, writable, user)
1021 <1> dpd_2:
1022 00004DC4 AB <1>      stosd
1023 00004DC5 E2E1 <1>      loop  dpd_0
1024 <1>      ;
1025 00004DC7 58 <1>      pop   eax ; restore child's page directory address
1026 <1> dpd_3:
1027 00004DC8 59 <1>      pop   ecx
1028 00004DC9 5B <1>      pop   ebx
1029 00004DCA 5F <1>      pop   edi
1030 00004DCB 5E <1>      pop   esi
1031 00004DCC 5D <1>      pop   ebp ; 20/07/2015
1032 <1> dpd_err:
1033 00004DCD C3 <1>      retn
1034 <1> dpd_p_err:
1035 <1>      ; release the allocated pages missing (recover free space)
1036 00004DCE 58 <1>      pop   eax ; the new page directory address (physical)
1037 00004DCF 8B1D[004F0100] <1>      mov   ebx, [u.pgdir] ; parent's page directory address
1038 00004DD5 E8D8FEFFFF <1>      call  deallocate_page_dir
1039 00004DDA 29C0 <1>      sub   eax, eax ; 0
1040 00004DDC F9 <1>      stc
1041 00004DDD EBE9 <1>      jmp   short dpd_3
1042 <1>
1043 <1> duplicate_page_table:
1044 <1>      ; 20/02/2017
1045 <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> ; | | | | | | |
1212 <1> ; | PAGE DIRECTORY TABLE BASE ADDRESS | reserved | C | W | rsrvd |
1213 <1> ; | | | | | | |
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> ; | | | | | | | | | | | |
1224 <1> ; | PAGE TABLE BASE ADDRESS 31..12 | AVL | G | 0 | D | A | C | W | / | / | P |
1225 <1> ; | | | | | | | | | | | |
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> ; | | | | | | | | | | | |
1244 <1> ; | PAGE FRAME BASE ADDRESS 31..12 | AVL | G | A | D | A | C | W | / | / | P |
1245 <1> ; | | | | | | | | | | | |
1246 <1> ; +-----+-----+-----+-----+
1247 <1> ;
1248 <1> ; P - PRESENT
1249 <1> ; R/W - READ/WRITE
1250 <1> ; U/S - USER/SUPERVISOR
1251 <1> ; PWT - WRITE THROUGH
1252 <1> ; PCD - CACHE DISABLE
1253 <1> ; A - ACCESSED
1254 <1> ; D - DIRTY
1255 <1> ; PAT - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
1256 <1> ; G - GLOBAL
1257 <1> ; AVL - AVAILABLE FOR SYSTEMS PROGRAMMER USE
1258 <1> ;

```


168

```

1361 00004EAB 0F20D1 <1> mov ecx, cr2 ; CR2 contains the virtual address
1362 <1> ; which has been caused to page fault
1363 <1> ;
1364 00004EAE C1E914 <1> shr ecx, 20 ; shift 20 bits right
1365 00004EB1 80E1FC <1> and cl, 0FCh ; mask lower 2 bits to get PDE offset
1366 <1> ;
1367 00004EB4 01CB <1> add ebx, ecx ; now, EBX points to the relevant page dir entry
1368 00004EB6 8B0B <1> mov ecx, [ebx] ; physical (base) address of the page table
1369 00004EB8 F6C101 <1> test cl, 1 ; check bit 0 is set (1) or not (0).
1370 00004EBB 740B <1> jz short pfh_set_pde ; Page directory entry is not valid,
1371 <1> ; set/validate page directory entry
1372 00004EBD 6681E100F0 <1> and cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
1373 00004EC2 89CB <1> mov ebx, ecx ; Physical address of the page table
1374 00004EC4 89C1 <1> mov ecx, eax ; new page address (physical)
1375 00004EC6 EB16 <1> jmp short pfh_get_pte
1376 <1> pfh_set_pde:
1377 <1> ; NOTE: Page directories and page tables never be swapped out!
1378 <1> ; (So, we know this PDE is empty or invalid)
1379 <1> ;
1380 00004EC8 08D0 <1> or al, dl ; lower 3 bits are used as U/S, R/W, P flags
1381 00004ECA 8903 <1> mov [ebx], eax ; Let's put the new page directory entry here !
1382 00004ECC 30C0 <1> xor al, al ; clear lower (3..8) bits
1383 00004ECE 89C3 <1> mov ebx, eax
1384 00004ED0 E8A4FCFFFF <1> call allocate_page ; (allocate a new page)
1385 00004ED5 7241 <1> jc short pfh_im_err ; 'insufficient memory' error
1386 <1> pfh_spde_1:
1387 <1> ; EAX = Physical (base) address of the allocated (new) page
1388 00004ED7 89C1 <1> mov ecx, eax
1389 00004ED9 E815FDFFFF <1> call clear_page ; Clear page content
1390 <1> pfh_get_pte:
1391 00004EDE 0F20D0 <1> mov eax, cr2 ; virtual address
1392 <1> ; which has been caused to page fault
1393 00004EE1 89C7 <1> mov edi, eax ; 20/07/2015
1394 00004EE3 C1E80C <1> shr eax, 12 ; shift 12 bit right to get
1395 <1> ; higher 20 bits of the page fault address
1396 00004EE6 25FF030000 <1> and eax, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
1397 00004EEB C1E002 <1> shl eax, 2 ; shift 2 bits left to get PTE offset
1398 00004EEE 01C3 <1> add ebx, eax ; now, EBX points to the relevant page table entry
1399 00004EF0 8B03 <1> mov eax, [ebx] ; get previous value of pte
1400 <1> ; bit 0 of EAX is always 0 (otherwise we would not be here)
1401 00004EF2 21C0 <1> and eax, eax
1402 00004EF4 7410 <1> jz short pfh_gpte_1
1403 <1> ; 20/07/2015
1404 00004EF6 87D9 <1> xchg ebx, ecx ; new page address (physical)
1405 00004EF8 55 <1> push ebp ; 20/07/2015
1406 00004EF9 0F20D5 <1> mov ebp, cr2
1407 <1> ; ECX = physical address of the page table entry
1408 <1> ; EBX = Memory page address (physical!)
1409 <1> ; EAX = Swap disk (offset) address
1410 <1> ; EBP = virtual address (page fault address)
1411 00004EFC E8B7000000 <1> call swap_in
1412 00004F01 5D <1> pop ebp
1413 00004F02 7210 <1> jc short pfh_err_retn
1414 00004F04 87CB <1> xchg ecx, ebx
1415 <1> ; EBX = physical address of the page table entry
1416 <1> ; ECX = new page
1417 <1> pfh_gpte_1:
1418 00004F06 08D1 <1> or cl, dl ; lower 3 bits are used as U/S, R/W, P flags
1419 00004F08 890B <1> mov [ebx], ecx ; Let's put the new page table entry here !
1420 <1> pfh_cpp_ok:
1421 <1> ; 20/07/2015
1422 00004F0A 0F20D3 <1> mov ebx, cr2
1423 00004F0D E8A6020000 <1> call add_to_swap_queue
1424 <1> ;
1425 <1> ; The new PTE (which contains the new page) will be added to
1426 <1> ; the swap queue, here.
1427 <1> ; (Later, if memory will become insufficient,
1428 <1> ; one page will be swapped out which is at the head of
1429 <1> ; the swap queue by using FIFO and access check methods.)
1430 <1> ;
1431 00004F12 31C0 <1> xor eax, eax ; 0
1432 <1> ;
1433 <1> pfh_err_retn:
1434 00004F14 59 <1> pop ecx
1435 00004F15 5A <1> pop edx
1436 00004F16 5B <1> pop ebx
1437 00004F17 C3 <1> retn
1438 <1>
1439 <1> pfh_im_err:
1440 00004F18 B8E4000000 <1> mov eax, ERR_MAJOR_PF + ERR_MINOR_IM ; Error code in AX
1441 <1> ; Major (Primary) Error: Page Fault
1442 <1> ; Minor (Secondary) Error: Insufficient Memory !
1443 00004F1D EBF5 <1> jmp short pfh_err_retn
1444 <1>
1445 <1>
1446 <1> pfh_p_err: ; 09/03/2015
1447 <1> pfh_pv_err:
1448 <1> ; Page fault was caused by a protection-violation
1449 00004F1F B8E6000000 <1> mov eax, ERR_MAJOR_PF + ERR_MINOR_PV ; Error code in AX
1450 <1> ; Major (Primary) Error: Page Fault
1451 <1> ; Minor (Secondary) Error: Protection violation !
1452 00004F24 F9 <1> stc
1453 00004F25 EBED <1> jmp short pfh_err_retn
1454 <1>
1455 <1> copy_page:
1456 <1> ; 22/09/2015
1457 <1> ; 21/09/2015
1458 <1> ; 19/09/2015
1459 <1> ; 07/09/2015
1460 <1> ; 31/08/2015
1461 <1> ; 20/07/2015
1462 <1> ; 05/05/2015

```

```

1463 <1> ; 03/05/2015
1464 <1> ; 18/04/2015
1465 <1> ; 12/04/2015
1466 <1> ; 30/10/2014
1467 <1> ; 18/10/2014 (Retro UNIX 386 v1 - beginning)
1468 <1> ;
1469 <1> ; INPUT ->
1470 <1> ; EBX = Virtual (linear) address of source page
1471 <1> ; (Page fault address)
1472 <1> ; OUTPUT ->
1473 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
1474 <1> ; (corresponding PAGE TABLE ENTRY is mapped/set)
1475 <1> ; EAX = 0 (CF = 1)
1476 <1> ; if there is not a free page to be allocated
1477 <1> ; (page content of the source page will be copied
1478 <1> ; onto the target/new page)
1479 <1> ;
1480 <1> ; Modified Registers -> ecx, ebx (except EAX)
1481 <1> ;
1482 00004F27 56 <1> push esi
1483 00004F28 57 <1> push edi
1484 <1> ;push ebx
1485 <1> ;push ecx
1486 00004F29 31F6 <1> xor esi, esi
1487 00004F2B C1EB0C <1> shr ebx, 12 ; shift 12 bits right to get PDE & PTE numbers
1488 00004F2E 89D9 <1> mov ecx, ebx ; save page fault address (as 12 bit shifted)
1489 00004F30 C1EB08 <1> shr ebx, 8 ; shift 8 bits right and then
1490 00004F33 80E3FC <1> and bl, 0FCh ; mask lower 2 bits to get PDE offset
1491 00004F36 89DF <1> mov edi, ebx ; save it for the parent of current process
1492 00004F38 031D[004F0100] <1> add ebx, [u.pgdir] ; EBX points to the relevant page dir entry
1493 00004F3E 8B03 <1> mov eax, [ebx] ; physical (base) address of the page table
1494 00004F40 662500F0 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
1495 00004F44 89CB <1> mov ebx, ecx ; (restore higher 20 bits of page fault address)
1496 00004F46 81E3FF030000 <1> and ebx, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
1497 00004F4C 66C1E302 <1> shl bx, 2 ; shift 2 bits left to get PTE offset
1498 00004F50 01C3 <1> add ebx, eax ; EBX points to the relevant page table entry
1499 <1> ; 07/09/2015
1500 00004F52 66F7030002 <1> test word [ebx], PTE_DUPLICATED ; (Does current process share this
1501 <1> ; read only page as a child process?)
1502 00004F57 7509 <1> jnz short cpp_0 ; yes
1503 00004F59 8B0B <1> mov ecx, [ebx] ; PTE value
1504 00004F5B 6681E100F0 <1> and cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
1505 00004F60 EB32 <1> jmp short cpp_1
1506 <1> cpp_0:
1507 00004F62 89FE <1> mov esi, edi
1508 00004F64 0335[044F0100] <1> add esi, [u.pgdir] ; the parent's page directory entry
1509 00004F6A 8B06 <1> mov eax, [esi] ; physical (base) address of the page table
1510 00004F6C 662500F0 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
1511 00004F70 89CE <1> mov esi, ecx ; (restore higher 20 bits of page fault address)
1512 00004F72 81E6FF030000 <1> and esi, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
1513 00004F78 66C1E602 <1> shl si, 2 ; shift 2 bits left to get PTE offset
1514 00004F7C 01C6 <1> add esi, eax ; EDX points to the relevant page table entry
1515 00004F7E 8B0E <1> mov ecx, [esi] ; PTE value of the parent process
1516 <1> ; 21/09/2015
1517 00004F80 8B03 <1> mov eax, [ebx] ; PTE value of the child process
1518 00004F82 662500F0 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear page attributes
1519 <1> ;
1520 00004F86 F6C101 <1> test cl, PTE_A_PRESENT ; is it a present/valid page ?
1521 00004F89 7424 <1> jz short cpp_3 ; the parent's page is not same page
1522 <1> ;
1523 00004F8B 6681E100F0 <1> and cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
1524 00004F90 39C8 <1> cmp eax, ecx ; Same page?
1525 00004F92 751B <1> jne short cpp_3 ; Parent page and child page are not same
1526 <1> ; Convert child's page to writable page
1527 <1> cpp_1:
1528 00004F94 E8E0FBFFFF <1> call allocate_page
1529 00004F99 721A <1> jc short cpp_4 ; 'insufficient memory' error
1530 00004F9B 21F6 <1> and esi, esi ; check ESI is valid or not
1531 00004F9D 7405 <1> jz short cpp_2
1532 <1> ; Convert read only page to writable page
1533 <1> ; (for the parent of the current process)
1534 <1> ;and word [esi], PTE_A_CLEAR ; 0F000h
1535 <1> ; 22/09/2015
1536 00004F9F 890E <1> mov [esi], ecx
1537 00004FA1 800E07 <1> or byte [esi], PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER
1538 <1> ; 1+2+4 = 7
1539 <1> cpp_2:
1540 00004FA4 89C7 <1> mov edi, eax ; new page address of the child process
1541 <1> ; 07/09/2015
1542 00004FA6 89CE <1> mov esi, ecx ; the page address of the parent process
1543 00004FA8 B900040000 <1> mov ecx, PAGE_SIZE / 4
1544 00004FAD F3A5 <1> rep movsd ; 31/08/2015
1545 <1> cpp_3:
1546 00004FAF 0C07 <1> or al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 1+2+4 = 7
1547 00004FB1 8903 <1> mov [ebx], eax ; Update PTE
1548 00004FB3 28C0 <1> sub al, al ; clear attributes
1549 <1> cpp_4:
1550 <1> ;pop ecx
1551 <1> ;pop ebx
1552 00004FB5 5F <1> pop edi
1553 00004FB6 5E <1> pop esi
1554 00004FB7 C3 <1> retn
1555 <1>
1556 <1> ;; 28/04/2015
1557 <1> ;; 24/10/2014
1558 <1> ;; 21/10/2014 (Retro UNIX 386 v1 - beginning)
1559 <1> ;; SWAP_PAGE_QUEUE (4096 bytes)
1560 <1> ;;
1561 <1> ;; 0000 0001 0002 0003 .... 1020 1021 1022 1023
1562 <1> ;; +-----+-----+-----+-----+-----+-----+
1563 <1> ;; | pg1 | pg2 | pg3 | pg4 | .... |pg1021|pg1022|pg1023|pg1024|
1564 <1> ;; +-----+-----+-----+-----+-----+-----+

```

```

1565 <1> ;;
1566 <1> ;; [swpq_last] = 0 to 4096 (step 4) -> the last position on the queue
1567 <1> ;;
1568 <1> ;; Method:
1569 <1> ;; Swap page queue is a list of allocated pages with physical
1570 <1> ;; addresses (system mode virtual addresses = physical addresses).
1571 <1> ;; It is used for 'swap_in' and 'swap_out' procedures.
1572 <1> ;; When a new page is being allocated, swap queue is updated
1573 <1> ;; by 'swap_queue_shift' procedure, header of the queue (offset 0)
1574 <1> ;; is checked for 'accessed' flag. If the 1st page on the queue
1575 <1> ;; is 'accessed' or 'read only', it is dropped from the list;
1576 <1> ;; other pages from the 2nd to the last (in [swpq_last]) shifted
1577 <1> ;; to head then the 2nd page becomes the 1st and '[swpq_last]'
1578 <1> ;; offset value becomes it's previous offset value - 4.
1579 <1> ;; If the 1st page of the swap page queue is not 'accessed'
1580 <1> ;; the queue/list is not shifted.
1581 <1> ;; After the queue/list shift, newly allocated page is added
1582 <1> ;; to the tail of the queue at the [swpq_count*4] position.
1583 <1> ;; But, if [swpq_count] > 1023, the newly allocated page
1584 <1> ;; will not be added to the tail of swap page queue.
1585 <1> ;;
1586 <1> ;; During 'swap_out' procedure, swap page queue is checked for
1587 <1> ;; the first non-accessed, writable page in the list,
1588 <1> ;; from the head to the tail. The list is shifted to left
1589 <1> ;; (to the head) till a non-accessed page will be found in the list.
1590 <1> ;; Then, this page is swapped out (to disk) and then it is dropped
1591 <1> ;; from the list by a final swap queue shift. [swpq_count] value
1592 <1> ;; is changed. If all pages on the queue are 'accessed',
1593 <1> ;; 'insufficient memory' error will be returned ('swap_out'
1594 <1> ;; procedure will be failed)...
1595 <1> ;;
1596 <1> ;; Note: If the 1st page of the queue is an 'accessed' page,
1597 <1> ;; 'accessed' flag of the page will be reset (0) and that page
1598 <1> ;; (PTE) will be added to the tail of the queue after
1599 <1> ;; the check, if [swpq_count] < 1023. If [swpq_count] = 1024
1600 <1> ;; the queue will be rotated and the PTE in the head will be
1601 <1> ;; added to the tail after resetting 'accessed' bit.
1602 <1> ;;
1603 <1> ;;
1604 <1> ;;
1605 <1> ;; SWAP DISK/FILE (with 4096 bytes swapped page blocks)
1606 <1> ;;
1607 <1> ;; 00000000 00000004 00000008 0000000C ... size-8 size-4
1608 <1> ;; +-----+-----+-----+-----+-----+-----+
1609 <1> ;; |descriptr| page(1) | page(2) | page(3) | ... |page(n-1)| page(n) |
1610 <1> ;; +-----+-----+-----+-----+-----+-----+
1611 <1> ;;
1612 <1> ;; [swpd_next] = the first free block address in swapped page records
1613 <1> ;; for next free block search by 'swap_out' procedure.
1614 <1> ;; [swpd_size] = swap disk/file size in sectors (512 bytes)
1615 <1> ;; NOTE: max. possible swap disk size is 1024 GB
1616 <1> ;; (entire swap space must be accessed by using
1617 <1> ;; 31 bit offset address)
1618 <1> ;; [swpd_free] = free block (4096 bytes) count in swap disk/file space
1619 <1> ;; [swpd_start] = absolute/start address of the swap disk/file
1620 <1> ;; 0 for file, or beginning sector of the swap partition
1621 <1> ;; [swp_drv] = logical drive description table addr. of swap disk/file
1622 <1> ;;
1623 <1> ;;
1624 <1> ;; Method:
1625 <1> ;; When the memory (ram) becomes insufficient, page allocation
1626 <1> ;; procedure swaps out a page from memory to the swap disk
1627 <1> ;; (partition) or swap file to get a new free page at the memory.
1628 <1> ;; Swapping out is performed by using swap page queue.
1629 <1> ;;
1630 <1> ;; Allocation block size of swap disk/file is equal to page size
1631 <1> ;; (4096 bytes). Swapping address (in sectors) is recorded
1632 <1> ;; into relevant page file entry as 31 bit physical (logical)
1633 <1> ;; offset address as 1 bit shifted to left for present flag (0).
1634 <1> ;; Swapped page address is between 1 and swap disk/file size - 4.
1635 <1> ;; Absolute physical (logical) address of the swapped page is
1636 <1> ;; calculated by adding offset value to the swap partition's
1637 <1> ;; start address. If the swap device (disk) is a virtual disk
1638 <1> ;; or it is a file, start address of the swap disk/volume is 0,
1639 <1> ;; and offset value is equal to absolute (physical or logical)
1640 <1> ;; address/position. (It has not to be ZERO if the swap partition
1641 <1> ;; is in a partitioned virtual hard disk.)
1642 <1> ;;
1643 <1> ;; Note: Swap addresses are always specified/declared in sectors,
1644 <1> ;; not in bytes or in blocks/zones/clusters (4096 bytes) as unit.
1645 <1> ;;
1646 <1> ;; Swap disk/file allocation is mapped via 'Swap Allocation Table'
1647 <1> ;; at memory as similar to 'Memory Allocation Table'.
1648 <1> ;;
1649 <1> ;; Every bit of Swap Allocation Table represents one swap block
1650 <1> ;; (equal to page size) respectively. Bit 0 of the S.A.T. byte 0
1651 <1> ;; is reserved for swap disk/file block 0 as descriptor block
1652 <1> ;; (also for compatibility with PTE). If bit value is ZERO,
1653 <1> ;; it means relevant (respective) block is in use, and,
1654 <1> ;; of course, if bit value is 1, it means relevant (respective)
1655 <1> ;; swap disk/file block is free.
1656 <1> ;; For example: bit 1 of the byte 128 represents block 1025
1657 <1> ;; (128*8+1) or sector (offset) 8200 on the swap disk or
1658 <1> ;; byte (offset/position) 4198400 in the swap file.
1659 <1> ;; 4GB swap space is represented via 128KB Swap Allocation Table.
1660 <1> ;; Initial layout of Swap Allocation Table is as follows:
1661 <1> ;; -----
1662 <1> ;; 01111111111111111111111111111111 .... 11111111111111111111111111111111
1663 <1> ;; -----
1664 <1> ;; (0 is reserved block, 1s represent free blocks respectively.)
1665 <1> ;; (Note: Allocation cell/unit of the table is bit, not byte)
1666 <1> ;;

```

```

1667 <1> ;; .....
1668 <1> ;;
1669 <1> ;; 'swap_out' procedure checks 'free_swap_blocks' count at first,
1670 <1> ;; then it searches Swap Allocation Table if free count is not
1671 <1> ;; zero. From begining the [swpd_next] dword value, the first bit
1672 <1> ;; position with value of 1 on the table is converted to swap
1673 <1> ;; disk/file offset address, in sectors (not 4096 bytes block).
1674 <1> ;; 'ldrv_write' procedure is called with ldrv (logical drive
1675 <1> ;; number of physical swap disk or virtual swap disk)
1676 <1> ;; number, sector offset (not absolute sector -LBA- number),
1677 <1> ;; and sector count (8, 512*8 = 4096) and buffer address
1678 <1> ;; (memory page). That will be a direct disk write procedure.
1679 <1> ;; (for preventing late memory allocation, significant waiting).
1680 <1> ;; If disk write procedure returns with error or free count of
1681 <1> ;; swap blocks is ZERO, 'swap_out' procedure will return with
1682 <1> ;; 'insufficient memory error' (cf=1).
1683 <1> ;;
1684 <1> ;; (Note: Even if free swap disk/file blocks was not zero,
1685 <1> ;; any disk write error will not be fixed by 'swap_out' procedure,
1686 <1> ;; in other words, 'swap_out' will not check the table for other
1687 <1> ;; free blocks after a disk write error. It will return to
1688 <1> ;; the caller with error (CF=1) which means swapping is failed.
1689 <1> ;;
1690 <1> ;; After writing the page on to swap disk/file address/sector,
1691 <1> ;; 'swap_out' procedure returns with that swap (offset) sector
1692 <1> ;; address (cf=0).
1693 <1> ;;
1694 <1> ;; .....
1695 <1> ;;
1696 <1> ;; 'swap_in' procedure loads addressed (relevant) swap disk or
1697 <1> ;; file sectors at specified memory page. Then page allocation
1698 <1> ;; procedure updates relevant page table entry with 'present'
1699 <1> ;; attribute. If swap disk or file reading fails there is nothing
1700 <1> ;; to do, except to terminate the process which is the owner of
1701 <1> ;; the swapped page.
1702 <1> ;;
1703 <1> ;; 'swap_in' procedure sets the relevant/respective bit value
1704 <1> ;; in the Swap Allocation Table (as free block). 'swap_in' also
1705 <1> ;; updates [swpd_first] pointer if it is required.
1706 <1> ;;
1707 <1> ;; .....
1708 <1> ;;
1709 <1> ;; Note: If [swap_enabled] value is ZERO, that means there is not
1710 <1> ;; a swap disk or swap file in use... 'swap_in' and 'swap_out'
1711 <1> ;; procedures and 'swap page que' procedures will not be active...
1712 <1> ;; 'Insufficient memory' error will be returned by 'swap_out'
1713 <1> ;; and 'general protection fault' will be returned by 'swap_in'
1714 <1> ;; procedure, if it is called mistakenly (a wrong value in a PTE).
1715 <1> ;;
1716 <1>
1717 <1> swap_in:
1718 <1> ; 31/08/2015
1719 <1> ; 20/07/2015
1720 <1> ; 28/04/2015
1721 <1> ; 18/04/2015
1722 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
1723 <1> ;
1724 <1> ; INPUT ->
1725 <1> ; EBX = PHYSICAL (real/flat) ADDRESS OF THE MEMORY PAGE
1726 <1> ; EBP = VIRTUAL (LINEAR) ADDRESS (page fault address)
1727 <1> ; EAX = Offset Address for the swapped page on the
1728 <1> ; swap disk or in the swap file.
1729 <1> ;
1730 <1> ; OUTPUT ->
1731 <1> ; EAX = 0 if loading at memory has been successful
1732 <1> ;
1733 <1> ; CF = 1 -> swap disk reading error (disk/file not present
1734 <1> ; or sector not present or drive not ready
1735 <1> ; EAX = Error code
1736 <1> ; [u.error] = EAX
1737 <1> ; = The last error code for the process
1738 <1> ; (will be reset after returning to user)
1739 <1> ;
1740 <1> ; Modified Registers -> EAX
1741 <1> ;
1742 <1>
1743 00004FB8 833D[AA500100]00 <1> cmp dword [swp_drv], 0
1744 00004FBF 7646 <1> jna short swpin_dnp_err
1745 <1>
1746 00004FC1 3B05[AE500100] <1> cmp eax, [swpd_size]
1747 00004FC7 734A <1> jnb short swpin_snp_err
1748 <1>
1749 00004FC9 56 <1> push esi
1750 00004FCA 53 <1> push ebx
1751 00004FCB 51 <1> push ecx
1752 00004FCC 8B35[AA500100] <1> mov esi, [swp_drv]
1753 00004FD2 B908000000 <1> mov ecx, PAGE_SIZE / LOGIC_SECT_SIZE ; 8 !
1754 <1> ; Note: Even if corresponding physical disk's sector
1755 <1> ; size different than 512 bytes, logical disk sector
1756 <1> ; size is 512 bytes and disk reading procedure
1757 <1> ; will be performed for reading 4096 bytes
1758 <1> ; (2*2048, 8*512).
1759 <1> ; ESI = Logical disk description table address
1760 <1> ; EBX = Memory page (buffer) address (physical!)
1761 <1> ; EAX = Sector address (offset address, logical sector number)
1762 <1> ; ECX = Sector count ; 8 sectors
1763 00004FD7 50 <1> push eax
1764 00004FD8 E8AF020000 <1> call logical_disk_read
1765 00004FDD 58 <1> pop eax
1766 00004FDE 730C <1> jnc short swpin_read_ok
1767 <1> ;
1768 00004FE0 B828000000 <1> mov eax, SWP_DISK_READ_ERR ; drive not ready or read error

```



```

1769 00004FE5 A3[104F0100] <1> mov [u.error], eax
1770 00004FEA EB17 <1> jmp short swpin_retn
1771 <1> ;
1772 <1> swpin_read_ok:
1773 <1> ; EAX = Offset address (logical sector number)
1774 00004FEC E80D020000 <1> call unlink_swap_block ; Deallocate swap block
1775 <1> ;
1776 <1> ; EBX = Memory page (buffer) address (physical!)
1777 <1> ; 20/07/2015
1778 00004FF1 89EB <1> mov ebx, ebp ; virtual address (page fault address)
1779 00004FF3 6681E300F0 <1> and bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
1780 00004FF8 8A1D[FB4E0100] <1> mov bl, [u.unol] ; current process number
1781 <1> ; EBX = Virtual (Linear) address & process number combination
1782 00004FFE E8DB000000 <1> call swap_queue_shift
1783 <1> ; eax = 0 ; 10/06/2016 (if ebx input > 0, eax output = 0)
1784 <1> ;sub eax, eax ; 0 ; Error Code = 0 (no error)
1785 <1> ; zf = 1
1786 <1> swpin_retn:
1787 00005003 59 <1> pop ecx
1788 00005004 5B <1> pop ebx
1789 00005005 5E <1> pop esi
1790 00005006 C3 <1> retn
1791 <1>
1792 <1> swpin_dnp_err:
1793 00005007 B829000000 <1> mov eax, SWP_DISK_NOT_PRESENT_ERR
1794 <1> swpin_err_retn:
1795 0000500C A3[104F0100] <1> mov [u.error], eax
1796 00005011 F9 <1> stc
1797 00005012 C3 <1> retn
1798 <1>
1799 <1> swpin_snp_err:
1800 00005013 B82A000000 <1> mov eax, SWP_SECTOR_NOT_PRESENT_ERR
1801 00005018 EBF2 <1> jmp short swpin_err_retn
1802 <1>
1803 <1> swap_out:
1804 <1> ; 10/06/2016
1805 <1> ; 07/06/2016
1806 <1> ; 23/05/2016
1807 <1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
1808 <1> ; 24/10/2014 - 31/08/2015 (Retro UNIX 386 v1)
1809 <1> ;
1810 <1> ; INPUT ->
1811 <1> ; none
1812 <1> ;
1813 <1> ; OUTPUT ->
1814 <1> ; EAX = Physical page address (which is swapped out
1815 <1> ; for allocating a new page)
1816 <1> ; CF = 1 -> swap disk writing error (disk/file not present
1817 <1> ; or sector not present or drive not ready
1818 <1> ; EAX = Error code
1819 <1> ; [u.error] = EAX
1820 <1> ; = The last error code for the process
1821 <1> ; (will be reset after returning to user)
1822 <1> ;
1823 <1> ; Modified Registers -> none (except EAX)
1824 <1> ;
1825 0000501A 66833D[A8500100]01 <1> cmp word [swpq_count], 1
1826 00005022 0F82AF000000 <1> jc swpout_im_err ; 'insufficient memory'
1827 <1>
1828 <1> ;cmp dword [swp_drv], 1
1829 <1> ;jc short swpout_dnp_err ; 'swap disk/file not present'
1830 <1>
1831 00005028 833D[B2500100]01 <1> cmp dword [swpd_free], 1
1832 0000502F 0F828F000000 <1> jc swpout_nfspc_err ; 'no free space on swap disk'
1833 <1>
1834 00005035 53 <1> push ebx ; *
1835 <1> swpout_1:
1836 <1> ; 10/06/2016
1837 00005036 31DB <1> xor ebx, ebx ; shift the queue and return a PTE value
1838 00005038 E8A1000000 <1> call swap_queue_shift
1839 0000503D 21C0 <1> and eax, eax ; 0 = empty queue (improper entries)
1840 0000503F 0F848A000000 <1> jz swpout_npts_err ; There is not any proper PTE
1841 <1> ; pointer in the swap queue
1842 <1> ; EAX = PTE value of the page
1843 <1> ; EBX = PTE address of the page
1844 00005045 662500F0 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
1845 <1> ;
1846 <1> ; 07/06/2016
1847 <1> ; 19/05/2016
1848 <1> ; check this page is in timer events or not
1849 <1>
1850 <1> swpout_timer_page_0:
1851 00005049 52 <1> push edx ; **
1852 <1>
1853 <1> ; 07/06/2016
1854 0000504A 803D[B7450100]00 <1> cmp byte [timer_events], 0
1855 00005051 762F <1> jna short swpout_2
1856 <1> ;
1857 00005053 8A15[B7450100] <1> mov dl, [timer_events]
1858 <1>
1859 00005059 51 <1> push ecx ; ***
1860 0000505A 53 <1> push ebx ; ****
1861 0000505B BB[A84F0100] <1> mov ebx, timer_set ; beginning address of timer event
1862 <1> ; structures
1863 <1> swpout_timer_page_1:
1864 00005060 8A0B <1> mov cl, [ebx]
1865 00005062 08C9 <1> or cl, cl ; 0 = free, >0 = process number
1866 00005064 7415 <1> jz short swpout_timer_page_3
1867 00005066 8B4B0C <1> mov ecx, [ebx+12] ; response (signal return) address
1868 00005069 6681E100F0 <1> and cx, PTE_A_CLEAR ; clear offset part (right 12 bits)
1869 <1> ; of the response byte address, to
1870 <1> ; get beginning of the page address)

```

```

1871 0000506E 39C8      <1>      cmp     eax, ecx
1872 00005070 7505      <1>      jne     short swpout_timer_page_2 ; not same page
1873                      <1>
1874                      <1>      ; !same page!
1875                      <1>      ;
1876                      <1>      ; NOTE: // 19/05/2016 // - TRDOS 386 feature only ! -
1877                      <1>      ; This page will be used by the kernel to put timer event
1878                      <1>      ; response (signal return) byte at the requested address;
1879                      <1>      ; in order to prevent a possible wrong write (while
1880                      <1>      ; this page is swapped out) on physical memory,
1881                      <1>      ; we must protect this page against to be swapped out!
1882                      <1>      ;
1883 00005072 5B         <1>      pop     ebx ; ****
1884 00005073 59         <1>      pop     ecx ; ***
1885 00005074 5A         <1>      pop     edx ; **
1886 00005075 EBBF      <1>      jmp     short swpout_1      ; do not swap out this page !
1887                      <1>
1888                      <1> swpout_timer_page_2:
1889                      <1>      ; 07/06/2016
1890 00005077 FECA      <1>      dec     dl
1891 00005079 7405      <1>      jz      short swpout_timer_page_4
1892                      <1> swpout_timer_page_3:
1893                      <1>      ;cmp     ebx, timer_set + 240 ; last timer event (15*16)
1894                      <1>      ;jnb     short swpout_timer_page_4
1895 0000507B 83C310     <1>      add     ebx, 16
1896 0000507E EBE0      <1>      jmp     short swpout_timer_page_1
1897                      <1>
1898                      <1> swpout_timer_page_4:
1899 00005080 5B         <1>      pop     ebx ; ****
1900 00005081 59         <1>      pop     ecx ; ***
1901                      <1> swpout_2:
1902 00005082 89DA      <1>      mov     edx, ebx      ; Page table entry address
1903 00005084 89C3      <1>      mov     ebx, eax      ; Buffer (Page) Address
1904                      <1>      ;
1905 00005086 E8A6010000 <1>      call    link_swap_block
1906 0000508B 7304      <1>      jnc     short swpout_3      ; It may not be needed here
1907                      <1>      ; because [swpd_free] value
1908                      <1>      ; was checked at the beginging.
1909 0000508D 5A         <1>      pop     edx ; **
1910 0000508E 5B         <1>      pop     ebx ; *
1911 0000508F EB33      <1>      jmp     short swpout_nfspc_err
1912                      <1> swpout_3:
1913 00005091 A900000080 <1>      test    eax, 80000000h ; test bit 31 (this may not be needed!)
1914 00005096 752C      <1>      jnz     short swpout_nfspc_err ; 10/06/2016 (bit 31 = 1 !)
1915                      <1>      ;
1916 00005098 56         <1>      push    esi ; **
1917 00005099 51         <1>      push    ecx ; ***
1918 0000509A 50         <1>      push    eax ; sector address ; (31 bit !, bit 31 = 0)
1919 0000509B 8B35[AA500100] <1>      mov     esi, [swp_drv]
1920 000050A1 B908000000 <1>      mov     ecx, PAGE_SIZE / LOGIC_SECT_SIZE ; 8 !
1921                      <1>      ; Note: Even if corresponding physical disk's sector
1922                      <1>      ; size different than 512 bytes, logical disk sector
1923                      <1>      ; size is 512 bytes and disk writing procedure
1924                      <1>      ; will be performed for writing 4096 bytes
1925                      <1>      ; (2*2048, 8*512).
1926                      <1>      ; ESI = Logical disk description table address
1927                      <1>      ; EBX = Buffer (Page) address
1928                      <1>      ; EAX = Sector address (offset address, logical sector number)
1929                      <1>      ; ECX = Sector count ; 8 sectors
1930                      <1>      ; edx = PTE address
1931 000050A6 E8E2010000 <1>      call    logical_disk_write
1932                      <1>      ; edx = PTE address
1933 000050AB 59         <1>      pop     ecx ; sector address
1934 000050AC 730C      <1>      jnc     short swpout_write_ok
1935                      <1>      ;
1936                      <1>      ; ; call      unlink_swap_block ; this block must be left as 'in use'
1937                      <1> swpout_dw_err:
1938 000050AE B82C000000 <1>      mov     eax, SWP_DISK_WRITE_ERR ; drive not ready or write error
1939 000050B3 A3[104F0100] <1>      mov     [u.error], eax
1940 000050B8 EB06      <1>      jmp     short swpout_retn
1941                      <1>      ;
1942                      <1> swpout_write_ok:
1943                      <1>      ; EBX = Buffer (page) address
1944                      <1>      ; EDX = Page Table Entry address
1945                      <1>      ; ECX = Swap disk sector (file block) address (31 bit)
1946 000050BA D1E1      <1>      shl     ecx, 1 ; 31 bit sector address from bit 1 to bit 31
1947 000050BC 890A      <1>      mov     [edx], ecx
1948                      <1>      ; bit 0 = 0 (swapped page)
1949 000050BE 89D8      <1>      mov     eax, ebx
1950                      <1> swpout_retn:
1951 000050C0 59         <1>      pop     ecx ; ***
1952 000050C1 5E         <1>      pop     esi ; **
1953 000050C2 5B         <1>      pop     ebx ; *
1954 000050C3 C3         <1>      retn
1955                      <1>
1956                      <1> ;swpout_dnp_err:
1957                      <1> ; mov     eax, SWP_DISK_NOT_PRESENT_ERR ; disk not present
1958                      <1> ; jmp     short swpout_err_retn
1959                      <1> swpout_nfspc_err:
1960 000050C4 B82B000000 <1>      mov     eax, SWP_NO_FREE_SPACE_ERR ; no free space
1961                      <1> swpout_err_retn:
1962 000050C9 A3[104F0100] <1>      mov     [u.error], eax
1963                      <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[A8500100] <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[A8500100] <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
2066 <1> ; ebx = virtual address (at page boundary)
2067 00005127 25FF0F0000 <1> and eax, PAGE_OFF ; 0FFFh
2068 <1> ; ax = process number (1 to 4095)
2069 0000512C 3A05[FB4E0100] <1> cmp al, [u.uno]
2070 <1> ; Max. 16 (nproc) processes for Retro UNIX 386 v1
2071 00005132 7507 <1> jne short swpqs_8
2072 00005134 A1[004F0100] <1> mov eax, [u.pgdir]
2073 00005139 EB28 <1> jmp short swpqs_9
2074 <1> swpqs_8:

```

```

2075                                     <1>      ; 09/06/2016
2076 0000513B 80B8[F54B0100]00          <1>      cmp     byte [eax+p.stat-1], 0
2077 00005142 76C4                      <1>      jna     short swpqs_3      ; free (or terminated) process
2078 00005144 80B8[F54B0100]02          <1>      cmp     byte [eax+p.stat-1], 2 ; waiting
2079 0000514B 77BB                      <1>      ja      short swpqs_3      ; zombie (3) or undefined ?
2080                                     <1>
2081                                     <1>      ;shl     ax, 2
2082 0000514D C0E002                     <1>      shl     al, 2
2083 00005150 8B80[024C0100]             <1>      mov     eax, [eax+p.upage-4]
2084 00005156 09C0                       <1>      or      eax, eax
2085 00005158 74AE                       <1>      jz      short swpqs_3 ; invalid upage
2086 0000515A 83C05C                     <1>      add     eax, u.pgdir - user
2087                                     <1>      ; u.pgdir value for the process
2088                                     <1>      ; is in [eax]
2089 0000515D 8B00                       <1>      mov     eax, [eax]
2090 0000515F 21C0                       <1>      and     eax, eax
2091 00005161 74A5                       <1>      jz      short swpqs_3 ; invalid page directory
2092                                     <1>      swpqs_9:
2093 00005163 52                         <1>      push    edx
2094                                     <1>      ; eax = page directory
2095                                     <1>      ; ebx = virtual address
2096 00005164 E82BFBFFFF                 <1>      call    get_pte
2097 00005169 89D3                       <1>      mov     ebx, edx      ; PTE address
2098 0000516B 5A                         <1>      pop     edx
2099                                     <1>      ; 10/06/2016
2100 0000516C 723A                       <1>      jc      short swpqs_13 ; empty PDE
2101                                     <1>      ; EAX = PTE value
2102 0000516E A801                       <1>      test    al, PTE_A_PRESENT ; bit 0 = 1
2103 00005170 7436                       <1>      jz      short swpqs_13 ; Drop non-present page
2104                                     <1>      ; from the queue (head)
2105 00005172 A802                       <1>      test    al, PTE_A_WRITE   ; bit 1 = 0 (read only)
2106 00005174 7432                       <1>      jz      short swpqs_13 ; Drop read only page
2107                                     <1>      ; from the queue (head)
2108                                     <1>      ;test   al, PTE_A_ACCESS ; bit 5 = 1 (Accessed)
2109                                     <1>      ;jnz    short swpqs_11 ; present
2110                                     <1>      ; accessed page
2111 00005176 0FBAF005                   <1>      btr     eax, PTE_A_ACCESS_BIT ; reset 'accessed' bit
2112 0000517A 7210                       <1>      jc      short swpqs_11 ; accessed page
2113                                     <1>
2114 0000517C 49                         <1>      dec     ecx
2115 0000517D 66890D[A8500100]           <1>      mov     [swpq_count], cx
2116 00005184 7402                       <1>      jz      short swpqs_10
2117                                     <1>      ; esi = head + 4
2118                                     <1>      ; edi = head
2119 00005186 F3A5                       <1>      rep     movsd    ; n = 1 to k-1, [n - 1] = [n]
2120                                     <1>      swpqs_10:
2121 00005188 890F                       <1>      mov     [edi], ecx ; 0
2122 0000518A EB8C                       <1>      jmp     short swpqs_6 ; 26/03/2017
2123                                     <1>
2124                                     <1>      swpqs_11:
2125 0000518C 8903                       <1>      mov     [ebx], eax      ; save changed attribute
2126                                     <1>      ; Rotation (head -> tail)
2127 0000518E 49                         <1>      dec     ecx      ; entry count -> last entry number
2128 0000518F 74F7                       <1>      jz      short swpqs_10
2129                                     <1>      ; esi = head + 4
2130                                     <1>      ; edi = head
2131 00005191 8B07                       <1>      mov     eax, [edi] ; 20/07/2015
2132 00005193 F3A5                       <1>      rep     movsd    ; n = 1 to k-1, [n - 1] = [n]
2133 00005195 8907                       <1>      mov     [edi], eax ; head -> tail ; [k] = [1]
2134                                     <1>
2135 00005197 668B0D[A8500100]           <1>      mov     cx, [swpq_count]
2136                                     <1>
2137                                     <1>      swpqs_12:
2138 0000519E BE00E00800                 <1>      mov     esi, swap_queue ; head
2139 000051A3 E974FFFFFF                 <1>      jmp     swpqs_7
2140                                     <1>
2141                                     <1>      swpqs_13:
2142 000051A8 49                         <1>      dec     ecx
2143 000051A9 66890D[A8500100]           <1>      mov     [swpq_count], cx
2144 000051B0 0F845EFFFFFF                 <1>      jz      swpqs_5
2145 000051B6 EBE6                       <1>      jmp     short swpqs_12
2146                                     <1>
2147                                     <1>      add_to_swap_queue:
2148                                     <1>      ; temporary - 16/09/2015
2149 000051B8 C3                         <1>      retn
2150                                     <1>      ; 20/02/2017
2151                                     <1>      ; 20/07/2015
2152                                     <1>      ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
2153                                     <1>      ;
2154                                     <1>      ; Adds new page to swap queue
2155                                     <1>      ; (page directories and page tables must not be added
2156                                     <1>      ; to swap queue)
2157                                     <1>      ;
2158                                     <1>      ; INPUT ->
2159                                     <1>      ;     EBX = Linear (Virtual) addr for current process
2160                                     <1>      ;     [u.uno]
2161                                     <1>      ;     20/02/2017
2162                                     <1>      ;     (Linear address = CORE + user's virtual address)
2163                                     <1>      ;
2164                                     <1>      ; OUTPUT ->
2165                                     <1>      ;     EAX = [swpq_count]
2166                                     <1>      ;     (after the PTE has been added)
2167                                     <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[FB4E0100]             <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[A8500100]	<1>	movzx	eax, word [swpq_count]
2179	000051D1 663D0004	<1>	cmp	ax, 1024
2180	000051D5 7205	<1>	jb	short atsqq_1
2181	000051D7 6629C0	<1>	sub	ax, ax
2182	000051DA 5B	<1>	pop	ebx
2183	000051DB C3	<1>	retn	
2184		<1>	atsqq_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 atsqq_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>	atsqq_2:	
2193	000051F1 6640	<1>	inc	ax
2194	000051F3 891E	<1>	mov	[esi], ebx ; Virtual address + [u.uno] combination
2195	000051F5 66A3[A8500100]	<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[B6500100]	<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[B6500100]	<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[B2500100]	<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		&		

```

2279 0000523A 031D[B6500100] <1> add ebx, [swpd_next] ; Free block searching starts from here
2280 <1> ; next_free_swap_block >> 5
2281 00005240 030D[BA500100] <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 81E90000D00 <1> sub ecx, swap_alloc_table
2307 0000525A 890D[B6500100] <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[B2500100] <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 81E90000D00 <1> sub ecx, swap_alloc_table
2324 00005273 890D[B6500100] <1> mov [swpd_next], ecx ; Set first free block searching start
2325 <1> ; address/offset (to the next)
2326 00005279 FF0D[B2500100] <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 address (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 address (offset address, logical sector number)
2367 <1> ; ECX = Sector count
2368 <1> ;
2369 0000528D C3 <1> retn
2370 <1>
2371 <1> get_physical_addr:
2372 <1> ; 26/03/2017
2373 <1> ; 20/02/2017
2374 <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
2375 <1> ; 18/10/2015
2376 <1> ; 29/07/2015
2377 <1> ; 20/07/2015
2378 <1> ; 04/06/2015
2379 <1> ; 20/05/2015
2380 <1> ; 28/04/2015

```

```

2381      <1>      ; 18/04/2015
2382      <1>      ; Get physical address
2383      <1>      ;      (allocates a new page for user if it is not present)
2384      <1>      ;
2385      <1>      ; (This subroutine is needed for mapping user's virtual
2386      <1>      ; (buffer) address to physical address (of the buffer).)
2387      <1>      ; ('sys write', 'sys read' system calls...)
2388      <1>      ;
2389      <1>      ; INPUT ->
2390      <1>      ;      EBX = virtual address
2391      <1>      ;      u.pgdir = page directory (physical) address
2392      <1>      ;
2393      <1>      ; OUTPUT ->
2394      <1>      ;      EAX = physical address
2395      <1>      ;      EBX = linear address
2396      <1>      ;      EDX = physical address of the page frame
2397      <1>      ;      (with attribute bits)
2398      <1>      ;      ECX = byte count within the page frame
2399      <1>      ;
2400      <1>      ; Modified Registers -> EAX, EBX, ECX, EDX
2401      <1>      ;
2402      <1>      add     ebx, CORE ; 18/10/2015
2403      <1>      get_physical_addr_x: ; 27/05/2016
2404      <1>      mov     eax, [u.pgdir]
2405      <1>      call    get_pte
2406      <1>      ; EDX = Page table entry address (if CF=0)
2407      <1>      ;      Page directory entry address (if CF=1)
2408      <1>      ;      (Bit 0 value is 0 if PT is not present)
2409      <1>      ; EAX = Page table entry value (page address)
2410      <1>      ;      CF = 1 -> PDE not present or invalid ?
2411      <1>      jnc     short gpa_1
2412      <1>      ;
2413      <1>      call    allocate_page
2414      <1>      jc      short gpa_im_err ; 'insufficient memory' error
2415      <1>      gpa_0:
2416      <1>      call    clear_page
2417      <1>      ; EAX = Physical (base) address of the allocated (new) page
2418      <1>      or      al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER ; 4+2+1 = 7
2419      <1>      ;      ; lower 3 bits are used as U/S, R/W, P flags
2420      <1>      ;      ; (user, writable, present page)
2421      <1>      mov     [edx], eax ; Let's put the new page directory entry here !
2422      <1>      mov     eax, [u.pgdir]
2423      <1>      call    get_pte
2424      <1>      jc      short gpa_im_err ; 'insufficient memory' error
2425      <1>      gpa_1:
2426      <1>      ; EAX = PTE value, EDX = PTE address
2427      <1>      test    al, PTE_A_PRESENT
2428      <1>      jnz     short gpa_3 ; 26/03/2017
2429      <1>      or      eax, eax
2430      <1>      jz      short gpa_7 ; Allocate a new page
2431      <1>      ; 20/07/2015
2432      <1>      push    ebp
2433      <1>      mov     ebp, ebx ; virtual (linear) address
2434      <1>      ; reload swapped page
2435      <1>      call    reload_page ; 28/04/2015
2436      <1>      pop     ebp
2437      <1>      jc      short gpa_retn
2438      <1>      gpa_2:
2439      <1>      ; 26/03/2017
2440      <1>      ; 20/02/2017
2441      <1>      ; If a page will contain a Signal Response Byte
2442      <1>      ; it must not be swapped out, because
2443      <1>      ; timer service or irq callback service
2444      <1>      ; will write a signal return/response byte
2445      <1>      ; directly by using physical address of Signal
2446      <1>      ; Response Byte.(Even if process is not running,
2447      <1>      ; or it is running with swapped out pages.)
2448      <1>      ;
2449      <1>      ; 'no_page_swap' will be set by 'systimer' or
2450      <1>      ; 'syscalbac' sistem functions/calls. (*)
2451      <1>      ;
2452      <1>      cmp     byte [no_page_swap], 0
2453      <1>      jna     short gpa_4 ; this page can be swapped out
2454      <1>      ; this page must not be swapped out
2455      <1>      ; but 'no_page_swap' must be reset here
2456      <1>      ; immediately for other callers (*)
2457      <1>      ; (otherwise, swap queue would not be long enough)
2458      <1>      call    gpa_8 ; 26/03/2017
2459      <1>      jmp     short gpa_5
2460      <1>      gpa_3:
2461      <1>      ; 26/03/2017
2462      <1>      cmp     byte [no_page_swap], 0
2463      <1>      jna     short gpa_6 ; this page can be swapped out
2464      <1>      call    gpa_8
2465      <1>      jmp     short gpa_6
2466      <1>
2467      <1>      gpa_im_err:
2468      <1>      mov     eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
2469      <1>      ; Major error = 0 (No protection fault)
2470      <1>      retn
2471      <1>      gpa_4:
2472      <1>      ; 20/07/2015
2473      <1>      ; 20/05/2015
2474      <1>      ; add this page to swap queue
2475      <1>      push    eax
2476      <1>      ; EBX = Linear (CORE+virtual) address ; 20/02/2017
2477      <1>      call    add_to_swap_queue
2478      <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 (0FFFFFFFh), 4095 (0FFFh) -> -4095
2496 00005312 81C100100000 <1>      add     ecx, PAGE_SIZE
2497 00005318 F8        <1>      cld
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[F64A0100]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[FB4E0100] <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                    <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[004F0100] <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[004F0100] <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>      jnb     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>
2717                <1>      allocate_memory_block:
2718                <1>      ; 03/04/2016
2719                <1>      ; 02/04/2016
2720                <1>      ; 01/04/2016
2721                <1>      ; 14/03/2016
2722                <1>      ; 13/03/2016
2723                <1>      ; 12/03/2016 (TRDOS 386 = TRDOS v2.0)
2724                <1>      ; Allocating contiguous memory pages (in the kernel's memory space)
2725                <1>      ;
2726                <1>      ; INPUT ->
2727                <1>      ;      EAX = Beginning address (physical)
2728                <1>      ;      EAX = 0 -> Allocate memory block from the first proper aperture
2729                <1>      ;      ECX = Number of bytes to be allocated
2730                <1>      ;
2731                <1>      ; OUTPUT ->
2732                <1>      ;      1) cf = 0 -> successful
2733                <1>      ;      EAX = Beginning (physical) address of the allocated memory block
2734                <1>      ;      ECX = Number of allocated bytes (rounded up to page borders)
2735                <1>      ;      2) cf = 1 -> unsuccessful
2736                <1>      ;      2.1) If EAX > 0 ->
2737                <1>      ;          (Number of requested pages is more than # of free pages
2738                <1>      ;          but contiguous free pages -the aperture- is not enough!)
2739                <1>      ;          EAX = Beginning address of available aperture
2740                <1>      ;          (one of all aperture with max. aperture size/length)
2741                <1>      ;          ECX = Size of available aperture (memory block) in bytes
2742                <1>      ;      2.2) If EAX = 0 -> Out of memory error
2743                <1>      ;          (number of free pages is less than requested number)
2744                <1>      ;          ECX = Total number of free bytes (free pages * 4096)
2745                <1>      ;          (It is not number of contiguous free bytes)
2746                <1>      ;
2747                <1>      ; (Modified Registers -> EAX, ECX)
2748                <1>      ;
2749                <1>      ; PURPOSE: Loading a file at memory for copying or running etc.
2750                <1>      ; If this procedure returns with cf is set, ECX contains maximum
2751                <1>      ; available space and EAX contains the beginning address of it.
2752                <1>      ; If EAX has zero, ECX contains total number of free bytes.
2753                <1>      ; If requested block has been successfully allocated (by rounding up to
2754                <1>      ; the last page border), it must be deallocated later by using
2755                <1>      ; 'deallocate_memory_block' procedure.
2756                <1>
2757 00005423 52      <1>      push    edx ; *
2758 00005424 BAFF0F0000 <1>      mov     edx, PAGE_SIZE - 1 ; 4095
2759 00005429 01D1    <1>      add     ecx, edx
2760 0000542B 01D0    <1>      add     eax, edx
2761 0000542D C1E90C <1>      shr     ecx, PAGE_SHIFT ; 12
2762                <1>
2763                <1>      ; ECX = number of contiguous pages to be allocated
2764 00005430 8B15[28380100] <1>      mov     edx, [free_pages]
2765 00005436 39D1    <1>      cmp     ecx, edx
2766 00005438 7775    <1>      ja      short amb_6
2767                <1>
2768 0000543A C1E80C <1>      shr     eax, PAGE_SHIFT ; 12
2769                <1>
2770 0000543D 89C2    <1>      mov     edx, eax ; page number
2771 0000543F C1EA03 <1>      shr     edx, 3 ; to get offset to M.A.T.
2772                <1>      ; (1 allocation bit = 1 page)
2773                <1>      ; (1 allocation bytes = 8 pages)
2774 00005442 80E2FC <1>      and     dl, 0FCh ; clear lower 2 bits
2775                <1>      ; (to get 32 bit position)
2776 00005445 53      <1>      push    ebx ; **
2777                <1>      amb_0:
2778 00005446 890D[E0440100] <1>      mov     [mem_ipg_count], ecx ; initial (reset) value of page count
2779 0000544C 890D[E4440100] <1>      mov     [mem_pg_count], ecx
2780 00005452 31C9    <1>      xor     ecx, ecx ; 0
2781 00005454 890D[E8440100] <1>      mov     [mem_aperture], ecx ; 0
2782 0000545A 890D[EC440100] <1>      mov     [mem_max_aperture], ecx ; 0
2783                <1>
2784 00005460 BB00001000 <1>      mov     ebx, MEM_ALLOC_TBL ; Memory Allocation Table address.
2785 00005465 3B15[2C380100] <1>      cmp     edx, [next_page] ; Is the beginning page address lower
2786                <1>      ; than the address in 'next_page' ?
2787                <1>      ; (the first/next free page of user space)
2788 0000546B 7208    <1>      jnb     short amb_1

```

```

2789 0000546D 3B15[30380100] <1>      cmp     edx, [last_page]      ; is the beginning page address higher
2790                                <1>                                ; than the address in 'last_page' ?
2791                                <1>                                ; (end of the memory)
2792 00005473 7606 <1>      jna     short amb_2        ; no
2793                                <1> amb_1:
2794 00005475 8B15[2C380100] <1>      mov     edx, [next_page]      ; yes (reset to the first page of user space)
2795                                <1> amb_2:
2796 0000547B 01D3 <1>      add     ebx, edx
2797                                <1>
2798 0000547D A3[F0440100] <1>      mov     [mem_pg_pos], eax      ; beginning page no (for curr. mem. aperture)
2799 00005482 A3[F4440100] <1>      mov     [mem_max_pg_pos], eax ; beginning page no for max. mem. aperture
2800                                <1>
2801 00005487 83E01F <1>      and     eax, 1Fh          ; lower 5 bits only (0 to 31)
2802                                <1>                                ; (allocation bit position)
2803 0000548A 744B <1>      jz      short amb_10      ; 0
2804                                <1> amb_3:
2805 0000548C 8B13 <1>      mov     edx, [ebx]
2806 0000548E 88C1 <1>      mov     cl, al ; 1 to 31
2807 00005490 D3EA <1>      shr     edx, cl
2808 00005492 89D0 <1>      mov     eax, edx
2809                                <1> amb_4:
2810 00005494 D1E8 <1>      shr     eax, 1 ; (***)
2811 00005496 7321 <1>      jnc     short amb_7
2812 00005498 FF05[E8440100] <1>      inc     dword [mem_aperture]
2813 0000549E FF0D[E4440100] <1>      dec     dword [mem_pg_count]
2814 000054A4 747B <1>      jz      short amb_15
2815                                <1> amb_5:
2816 000054A6 80F91F <1>      cmp     cl, 31
2817 000054A9 7317 <1>      jnb     short amb_8
2818 000054AB FEC1 <1>      inc     cl
2819 000054AD EBE5 <1>      jmp     short amb_4
2820                                <1>
2821                                <1> amb_6:      ; out_of_memory
2822 000054AF 31C0 <1>      xor     eax, eax ; 0
2823 000054B1 89D1 <1>      mov     ecx, edx ; free pages
2824 000054B3 C1E10C <1>      shl     ecx, PAGE_SHIFT
2825 000054B6 5A <1>      pop     edx ; *
2826 000054B7 F9 <1>      stc
2827 000054B8 C3 <1>      retn
2828                                <1> amb_7:
2829 000054B9 50 <1>      push    eax ; (***) allocation bits (in shifted status)
2830 000054BA E819010000 <1>      call   amb_26 ; set maximum memory aperture (free memory block size)
2831 000054BF 58 <1>      pop     eax ; (***)
2832 000054C0 EBE4 <1>      jmp     short amb_5
2833                                <1> amb_8:
2834 000054C2 28C9 <1>      sub     cl, cl ; 0
2835                                <1> amb_9:
2836 000054C4 89DA <1>      mov     edx, ebx
2837 000054C6 81EA00001000 <1>      sub     edx, MEM_ALLOC_TBL
2838 000054CC 3B15[30380100] <1>      cmp     edx, [last_page]
2839 000054D2 7336 <1>      jnb     short amb_14 ; contiguous pages not enough
2840 000054D4 83C304 <1>      add     ebx, 4
2841                                <1> amb_10:
2842 000054D7 8B03 <1>      mov     eax, [ebx]
2843 000054D9 21C0 <1>      and     eax, eax
2844 000054DB 7406 <1>      jz      short amb_11 ; there is not a free page bit in this alloc dword
2845 000054DD 40 <1>      inc     eax ; 0FFFFFFFFh -> 0
2846 000054DE 740A <1>      jz      short amb_12 ; all of bits are set (32 free pages)
2847 000054E0 48 <1>      dec     eax
2848 000054E1 EBB1 <1>      jmp     short amb_4
2849                                <1> amb_11:
2850 000054E3 E8F0000000 <1>      call   amb_26 ; set maximum memory aperture (free memory block size)
2851 000054E8 EBDA <1>      jmp     short amb_9
2852                                <1> amb_12:
2853 000054EA B120 <1>      mov     cl, 32
2854 000054EC 390D[E4440100] <1>      cmp     [mem_pg_count], ecx ; 32
2855 000054F2 7306 <1>      jnb     short amb_13
2856 000054F4 8B0D[E4440100] <1>      mov     ecx, [mem_pg_count]
2857                                <1> amb_13:
2858 000054FA 010D[E8440100] <1>      add     [mem_aperture], ecx
2859 00005500 290D[E4440100] <1>      sub     [mem_pg_count], ecx
2860 00005506 7619 <1>      jna     short amb_15
2861 00005508 EBB8 <1>      jmp     short amb_8
2862                                <1> amb_14:
2863 0000550A A1[F4440100] <1>      mov     eax, [mem_max_pg_pos] ; begin address of max. mem aperture
2864 0000550F 8B0D[EC440100] <1>      mov     ecx, [mem_max_aperture] ; max. (largest) memory aperture
2865 00005515 C1E00C <1>      shl     eax, PAGE_SHIFT      ; convert to phy. address in bytes
2866 00005518 C1E10C <1>      shl     ecx, PAGE_SHIFT      ; convert to byte counts
2867 0000551B F9 <1>      stc
2868 0000551C E9AC000000 <1>      jmp     amb_25
2869                                <1>
2870                                <1> amb_15: ; OK !
2871 00005521 A1[F0440100] <1>      mov     eax, [mem_pg_pos]      ; Beginning address as page number
2872 00005526 8B0D[E8440100] <1>      mov     ecx, [mem_aperture]    ; Free contiguous page count
2873                                <1> amb_16:
2874                                <1>      ; allocate contiguous memory pages (via memory allocation table bits)
2875 0000552C 89C2 <1>      mov     edx, eax
2876 0000552E C1EA05 <1>      shr     edx, 5 ; 32 pages in one allocation dword (32 bits)
2877 00005531 BB00001000 <1>      mov     ebx, MEM_ALLOC_TBL
2878 00005536 01D3 <1>      add     ebx, edx
2879 00005538 83E01F <1>      and     eax, 1Fh ; 31
2880                                <1>      ; 03/04/2016
2881 0000553B BA20000000 <1>      mov     edx, 32
2882 00005540 28C2 <1>      sub     dl, al
2883 00005542 39CA <1>      cmp     edx, ecx
2884 00005544 7602 <1>      jna     short amb_17
2885 00005546 89CA <1>      mov     edx, ecx
2886                                <1> amb_17:
2887 00005548 29D1 <1>      sub     ecx, edx
2888 0000554A 51 <1>      push    ecx ; ***
2889 0000554B 89D1 <1>      mov     ecx, edx
2890                                <1> amb_18:

```

```

2891 0000554D 0FB303      <1>      btr      [ebx], eax      ; The destination bit indexed by the source value
2892                                <1>                                ; is copied into the Carry Flag and then cleared
2893                                <1>                                ; in the destination.
2894 00005550 FF0D[28380100] <1>      dec      dword [free_pages] ; 1 page has been allocated (X = X-1)
2895 00005556 49          <1>      dec      ecx
2896 00005557 7404          <1>      jz       short amb_19
2897 00005559 FEC0          <1>      inc      al
2898 0000555B EBF0          <1>      jmp      short amb_18
2899                                <1> amb_19:
2900 0000555D 59          <1>      pop      ecx ; ***
2901 0000555E 21C9          <1>      and      ecx, ecx ; 0 ?
2902 00005560 741E          <1>      jz       short amb_22
2903                                <1>      ; 01/04/2016
2904 00005562 B020          <1>      mov      al, 32
2905                                <1> amb_20:
2906 00005564 83C304          <1>      add      ebx, 4
2907 00005567 39C1          <1>      cmp      ecx, eax ; 32
2908 00005569 7305          <1>      jnb      short amb_21
2909                                <1>      ; ECX < 32
2910 0000556B 28C0          <1>      sub      al, al ; 0
2911 0000556D 50          <1>      push     eax ; 0 ***
2912 0000556E EBDD          <1>      jmp      short amb_18
2913                                <1> amb_21:
2914 00005570 2905[28380100] <1>      sub      [free_pages], eax ; [free_pages] = [free_pages] - 32
2915 00005576 C70300000000 <1>      mov      dword [ebx], 0 ; reset 32 bits
2916 0000557C 29C1          <1>      sub      ecx, eax ; 32
2917 0000557E 75E4          <1>      jnz      short amb_20
2918                                <1> amb_22:
2919 00005580 A1[F0440100] <1>      mov      eax, [mem_pg_pos] ; Beginning address as page number
2920 00005585 8B0D[E8440100] <1>      mov      ecx, [mem_aperture] ; Free contiguous page count
2921                                <1>      ; [next_page] update
2922 0000558B 89C2          <1>      mov      edx, eax
2923                                <1>      ; 03/04/2016
2924 0000558D C1EA03          <1>      shr      edx, 3      ; to get offset to M.A.T.
2925                                <1>                                ; (1 allocation bit = 1 page)
2926                                <1>                                ; (1 allocation bytes = 8 pages)
2927 00005590 80E2FC          <1>      and      dl, 0FCh      ; clear lower 2 bits
2928                                <1>                                ; (to get 32 bit position)
2929 00005593 3B15[2C380100] <1>      cmp      edx, [next_page] ; first free page pointer offset
2930 00005599 7732          <1>      ja       short amb_25
2931 0000559B BB00001000 <1>      mov      ebx, MEM_ALLOC_TBL
2932 000055A0 833C1300 <1>      cmp      dword [ebx+edx], 0
2933 000055A4 7721          <1>      ja       short amb_24
2934 000055A6 89C2          <1>      mov      edx, eax
2935 000055A8 01CA          <1>      add      edx, ecx
2936 000055AA C1EA03          <1>      shr      edx, 3
2937 000055AD 80E2FC          <1>      and      dl, 0FCh
2938                                <1> amb_23:
2939 000055B0 833C1300 <1>      cmp      dword [ebx+edx], 0
2940 000055B4 7711          <1>      ja       short amb_24
2941 000055B6 83C204          <1>      add      edx, 4
2942 000055B9 3B15[30380100] <1>      cmp      edx, [last_page] ; last page pointer offset
2943 000055BF 76EF          <1>      jna      short amb_23
2944 000055C1 8B15[34380100] <1>      mov      edx, [first_page] ; (for) beginning of user's space
2945                                <1> amb_24:
2946 000055C7 8915[2C380100] <1>      mov      [next_page], edx
2947                                <1> amb_25:
2948 000055CD 9C          <1>      pushf
2949 000055CE C1E00C          <1>      shl      eax, PAGE_SHIFT      ; convert to phy. address in bytes
2950 000055D1 C1E10C          <1>      shl      ecx, PAGE_SHIFT      ; convert to byte counts
2951 000055D4 9D          <1>      popf
2952 000055D5 5B          <1>      pop      ebx ; **
2953 000055D6 5A          <1>      pop      edx ; *
2954 000055D7 C3          <1>      retn
2955                                <1>
2956                                <1> amb_26:      ; set maximum free memory aperture (free memory block size)
2957 000055D8 89DA          <1>      mov      edx, ebx ; current address
2958 000055DA 81EA00001000 <1>      sub      edx, MEM_ALLOC_TBL ; MAT beginning address
2959                                <1>      ; 02/04/2016
2960 000055E0 C1E203          <1>      shl      edx, 3 ; MAT byte offset * 8 = page number base
2961 000055E3 01CA          <1>      add      edx, ecx ; current page number (ecx = 0 to 31)
2962                                <1>      ;
2963 000055E5 A1[E8440100] <1>      mov      eax, [mem_aperture]
2964 000055EA 21C0          <1>      and      eax, eax
2965 000055EC 7424          <1>      jz       short amb_27
2966 000055EE C705[E8440100]0000- <1>      mov      dword [mem_aperture], 0
2966 000055F6 0000          <1>
2967 000055F8 3B05[EC440100] <1>      cmp      eax, [mem_max_aperture]
2968 000055FE 7612          <1>      jna      short amb_27
2969 00005600 A3[EC440100] <1>      mov      [mem_max_aperture], eax
2970                                <1>      ;
2971 00005605 89D0          <1>      mov      eax, edx
2972 00005607 2B05[EC440100] <1>      sub      eax, [mem_max_aperture] ; the last aperture size in pages
2973                                <1>      ; EAX = Beginning page number of the max. aperture
2974 0000560D A3[F4440100] <1>      mov      [mem_max_pg_pos], eax
2975                                <1> amb_27:
2976 00005612 42          <1>      inc      edx
2977 00005613 8915[F0440100] <1>      mov      [mem_pg_pos], edx ; next page
2978                                <1>
2979 00005619 A1[E0440100] <1>      mov      eax, [mem_ipg_count] ; initial (reset) value of page count
2980 0000561E A3[E4440100] <1>      mov      [mem_pg_count], eax
2981                                <1>
2982 00005623 C3          <1>      retn
2983                                <1>
2984                                <1> deallocate_memory_block:
2985                                <1>      ; 03/04/2016
2986                                <1>      ; 14/03/2016 (TRDOS 386 = TRDOS v2.0)
2987                                <1>      ; Deallocating contiguous memory pages (in the kernel's memory space)
2988                                <1>      ;
2989                                <1>      ; INPUT ->
2990                                <1>      ;      EAX = Beginning address (physical)
2991                                <1>      ;      ECX = Number of bytes to be deallocated

```



```

2992      <1>      ;
2993      <1>      ; OUTPUT ->
2994      <1>      ;      Memory Allocation Table bits will be updated
2995      <1>      ;      [free_pages] will be changed (increased)
2996      <1>      ;
2997      <1>      ; (Modified Registers -> EAX, ECX)
2998      <1>      ;
2999      <1>      ; PURPOSE: Unloading/Freeing a file -or an allocated memory block-
3000      <1>      ; at memory after copying, running, saving, reading, writing etc.
3001      <1>      ;
3002      <1>
3003      00005624 52      <1>      push    edx ; *
3004      00005625 53      <1>      push    ebx ; **
3005      <1>
3006      00005626 C1E80C  <1>      shr     eax, PAGE_SHIFT      ; 12
3007      00005629 C1E90C  <1>      shr     ecx, PAGE_SHIFT      ; 12
3008      <1>
3009      <1>      ; EAX = Beginning page number
3010      <1>      ; ECX = Number of contiguous pages to be deallocated
3011      <1>      damb_0:
3012      <1>      ; deallocate contiguous memory pages (via memory allocation table bits)
3013      0000562C 89C2     <1>      mov     edx, eax
3014      0000562E C1EA03  <1>      shr     edx, 3              ; to get offset to M.A.T.
3015      <1>      ; (1 allocation bit = 1 page)
3016      <1>      ; (1 allocation bytes = 8 pages)
3017      00005631 80E2FC  <1>      and     dl, 0FCh          ; clear lower 2 bits
3018      <1>      ; (to get 32 bit position)
3019      00005634 3B15[2C380100] <1>      cmp     edx, [next_page] ; next free page
3020      0000563A 7306     <1>      jnb     short damb_1
3021      0000563C 8915[2C380100] <1>      mov     [next_page], edx
3022      <1>      damb_1:
3023      00005642 BB00001000 <1>      mov     ebx, MEM_ALLOC_TBL
3024      00005647 01D3     <1>      add     ebx, edx
3025      00005649 83E01F  <1>      and     eax, 1Fh ; 31
3026      <1>
3027      <1>      ; 03/04/2016
3028      0000564C BA20000000 <1>      mov     edx, 32
3029      00005651 28C2     <1>      sub     dl, al
3030      00005653 39CA     <1>      cmp     edx, ecx
3031      00005655 7602     <1>      jna     short damb_2
3032      00005657 89CA     <1>      mov     edx, ecx
3033      <1>      damb_2:
3034      00005659 29D1     <1>      sub     ecx, edx
3035      0000565B 51       <1>      push    ecx ; ***
3036      0000565C 89D1     <1>      mov     ecx, edx
3037      <1>      damb_3:
3038      0000565E 0FAB03  <1>      bts     [ebx], eax      ; unlink/release/deallocate page
3039      <1>      ; set relevant bit to 1.
3040      <1>      ; set CF to the previous bit value
3041      00005661 FF05[28380100] <1>      inc     dword [free_pages] ; 1 page has been deallocated (X = X+1)
3042      00005667 49       <1>      dec     ecx
3043      00005668 7404     <1>      jz      short damb_4
3044      0000566A FEC0     <1>      inc     al
3045      0000566C EBF0     <1>      jmp     short damb_3
3046      <1>      damb_4:
3047      0000566E 59       <1>      pop     ecx ; ***
3048      0000566F 21C9     <1>      and     ecx, ecx ; 0 ?
3049      00005671 741E     <1>      jz      short damb_7
3050      <1>      ; 03/04/2016
3051      00005673 B020     <1>      mov     al, 32
3052      <1>      damb_5:
3053      00005675 83C304  <1>      add     ebx, 4
3054      00005678 39C1     <1>      cmp     ecx, eax ; 32
3055      0000567A 7305     <1>      jnb     short damb_6
3056      <1>      ; ECX < 32
3057      0000567C 28C0     <1>      sub     al, al ; 0
3058      0000567E 50       <1>      push    eax ; 0 ***
3059      0000567F EBDD     <1>      jmp     short damb_3
3060      <1>      damb_6:
3061      00005681 0105[28380100] <1>      add     [free_pages], eax ; [free_pages] = [free_pages] + 32
3062      00005687 C703FFFFFFFFF <1>      mov     dword [ebx], 0FFFFFFFFh ; set 32 bits
3063      0000568D 29C1     <1>      sub     ecx, eax ; 32
3064      0000568F 75E4     <1>      jnz     short damb_5
3065      <1>      damb_7:
3066      00005691 5B       <1>      pop     ebx ; **
3067      00005692 5A       <1>      pop     edx ; *
3068      00005693 C3       <1>      retn
3069      <1>
3070      <1>      direct_memory_access:
3071      <1>      ; 16/07/2016
3072      <1>      ; 12/07/2016 (TRDOS 386 = TRDOS v2.0)
3073      <1>      ; This processure will be called to map
3074      <1>      ; user's (ring 3) page tables to access phsical
3075      <1>      ; (flat/linear) memory addresses, directly (without
3076      <1>      ; kernel's data transfer functions).
3077      <1>      ;
3078      <1>      ; Purpose: Video memory access and shared memory access.
3079      <1>      ;
3080      <1>      ; INPUT ->
3081      <1>      ;      EAX = Beginning address (physical).
3082      <1>      ;      ECX = Number of contiguous pages to be mapped.
3083      <1>      ; OUTPUT ->
3084      <1>      ;      User's page directory and pages tables
3085      <1>      ;      will be updated.
3086      <1>      ;
3087      <1>      ;      If an old page table entry has valid page address,
3088      <1>      ;      that page will be deallocated just before PTE will
3089      <1>      ;      be changed for direct (1 to 1) memory page access.
3090      <1>      ;
3091      <1>      ;      If old PTE value points to a swapped page,
3092      <1>      ;      that page (block) will be unlinked on swap disk.
3093      <1>      ;

```

```

3094      <1>      ;      Newly allocated pages (except page tables) will not
3095      <1>      ;      be applied to Memory Allocation Table.
3096      <1>      ;      AVL bit 1 (PTE bit 10) of page table entry will be
3097      <1>      ;      used to indicate shared (direct) memory page; then,
3098      <1>      ;      this page will not be deallocated later during
3099      <1>      ;      process termination. (Memory Allocation Table and
3100      <1>      ;      free memory count will not be affected.
3101      <1>      ;      (Except deallocating page table's itself.)
3102      <1>      ;
3103      <1>      ;      CF = 1 -> error (EAX = error code)
3104      <1>      ;      CF = 0 -> success (EAX = beginning address)
3105      <1>      ;
3106      <1>      ;; (Modified Registers -> none)
3107      <1>      ; Modified registers: ebp, edx, ecx, ebx, esi, edi
3108      <1>      ;
3109      <1>
3110      <1>      ;push  ebp
3111      <1>      ;push  ebx
3112      <1>      ;push  ecx
3113      <1>      ;push  edx
3114      <1>      and    ax, PTE_A_CLEAR ; clear page offset
3115      <1>      push  eax
3116      <1>      mov    ebp, eax
3117      <1> dmem_acc_0:
3118      <1>      mov    ebx, eax
3119      <1>      add    ebx, CORE
3120      <1>      mov    eax, [u.pgdir] ; page dir address (physical)
3121      <1>      call   get_pte
3122      <1>      ; EDX = Page table entry address (if CF=0)
3123      <1>      ;      Page directory entry address (if CF=1)
3124      <1>      ;      (Bit 0 value is 0 if PT is not present)
3125      <1>      ; EAX = Page table entry value (page address)
3126      <1>      ;      CF = 1 -> PDE not present or invalid ?
3127      <1>      jnc    short dmem_acc_1
3128      <1>      ;
3129      <1>      call   allocate_page
3130      <1>      jc     dmem_acc_7 ; 'insufficient memory' error
3131      <1>      ;
3132      <1>      call   clear_page
3133      <1>      ; EAX = Physical (base) address of the allocated (new) page
3134      <1>      or     al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER ; 4+2+1 = 7
3135      <1>      ;      lower 3 bits are used as U/S, R/W, P flags
3136      <1>      ;      (user, writable, present page)
3137      <1>      mov    [edx], eax ; Let's put the new page directory entry here !
3138      <1>      mov    eax, [u.pgdir]
3139      <1>      call   get_pte
3140      <1>      jc     dmem_acc_7 ; 'insufficient memory' error
3141      <1> dmem_acc_1:
3142      <1>      ; EAX = PTE value, EDX = PTE address
3143      <1>      test   al, PTE_A_PRESENT
3144      <1>      jnz    short dmem_acc_2
3145      <1>      or     eax, eax
3146      <1>      jz     short dmem_acc_6 ; Change PTE
3147      <1>      shr    eax, 1 ; swap disk block (8 sectors) address
3148      <1>      ; unlink swap disk block
3149      <1>      call   unlink_swap_block
3150      <1>      jmp    short dmem_acc_6
3151      <1>
3152      <1> dmem_acc_2:
3153      <1>      test   al, PTE_A_WRITE ; bit 1, writable (r/w) flag
3154      <1>      ;      (must be 1)
3155      <1>      jnz    short dmem_acc_4
3156      <1>      ; Read only -duplicated- page (belongs to a parent or a child)
3157      <1>      test   ax, PTE_DUPLICATED ; Was this page duplicated
3158      <1>      ;      as child's page ?
3159      <1>      jz     short dmem_acc_5 ; Change PTE but don't deallocate the page!
3160      <1>
3161      <1>      ;push  edi
3162      <1>      ;push  esi
3163      <1>
3164      <1>      push  ecx
3165      <1>      ;push  ebx
3166      <1>      mov    ebx, [u.ppgdir] ; parent's page dir address (physical)
3167      <1>
3168      <1>      ; check the parent's PTE value is read only & same page or not..
3169      <1>      mov    edi, ebp
3170      <1>      shr    edi, PAGE_D_SHIFT ; 22
3171      <1>      ; EDI = page directory entry index (0-1023)
3172      <1>      mov    esi, ebp
3173      <1>      shr    esi, PAGE_SHIFT ; 12
3174      <1>      and    esi, PTE_MASK
3175      <1>      ; ESI = page table entry index (0-1023)
3176      <1>
3177      <1>      shl    di, 2 ; * 4
3178      <1>      add    ebx, edi ; PDE offset (for the parent)
3179      <1>      mov    ecx, [edi]
3180      <1>      test   cl, PDE_A_PRESENT ; present (valid) or not ?
3181      <1>      jz     short dmem_acc_3 ; parent process does not use this page
3182      <1>      and    cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3183      <1>      shl    si, 2 ; * 4
3184      <1>      add    esi, ecx ; PTE offset (for the parent)
3185      <1>      mov    ebx, [esi]
3186      <1>      test   bl, PTE_A_PRESENT ; present or not ?
3187      <1>      jz     short dmem_acc_3 ; parent process does not use this page
3188      <1>      and    ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3189      <1>      and    bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3190      <1>      cmp    eax, ebx ; parent's and child's pages are same ?
3191      <1>      jne    short dmem_acc_3 ; not same page
3192      <1>      ; deallocate the child's page
3193      <1>      or     byte [esi], PTE_A_WRITE ; convert to writable page (parent)
3194      <1>      ;pop   ebx
3195      <1>      pop    ecx

```

```

3196 00005735 EB0C      <1>      jmp      short dmem_acc_5
3197                    <1> dmem_acc_3:
3198                    <1>      ;pop      ebx
3199 00005737 59        <1>      pop      ecx
3200                    <1> dmem_acc_4:
3201 00005738 66A90004   <1>      test     ax, PTE_SHARED ; shared or direct memory access indicator
3202 0000573C 7505       <1>      jnz      short dmem_acc_5 ; AVL bit 1 = 1, do not deallocate this page!
3203                    <1>      ;
3204                    <1>      ;and     ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3205 0000573E E814F6FFFF   <1>      call     deallocate_page
3206                    <1> dmem_acc_5:
3207                    <1>      ;pop      esi
3208                    <1>      ;pop      edi
3209                    <1> dmem_acc_6:
3210 00005743 89E8       <1>      mov      eax, ebp ; physical page (offset=0) address
3211                    <1>      ; EAX = memory page address
3212                    <1>      ; EDX = PTE entry address (physical)
3213 00005745 660D0704   <1>      or       ax, PTE_A_PRESENT+PTE_A_USER+PTE_A_WRITE+PTE_SHARED
3214                    <1>      ; present flag, bit 0 = 1
3215                    <1>      ; user flag, bit 2 = 1
3216                    <1>      ; writable flag, bit 1 = 1
3217                    <1>      ; direct memory access flag, bit 10 = 1
3218                    <1>      ; (This page must not be deallocated!)
3219 00005749 8902       <1>      mov      [edx], eax ; Update PTE value
3220 0000574B 49         <1>      dec      ecx ; remain count of contiguous pages
3221 0000574C 7414       <1>      jz       short dmem_acc_8
3222 0000574E 81C500100000 <1>      add      ebp, PAGE_SIZE ; next virtual (=physical) page address
3223 00005754 89E8       <1>      mov      eax, ebp
3224 00005756 E940FFFFFF   <1>      jmp      dmem_acc_0
3225                    <1> dmem_acc_7: ; ERROR !
3226 0000575B C7042404000000 <1>      mov      dword [esp], ERR_MINOR_IM
3227                    <1>      ; Insufficient memory (minor) error!
3228                    <1>      ; Major error = 0 (No protection fault)
3229                    <1>      ; cf = 1
3230                    <1> dmem_acc_8:
3231 00005762 58         <1>      pop      eax
3232                    <1>      ;pop      edx
3233                    <1>      ;pop      ecx
3234                    <1>      ;pop      ebx
3235                    <1>      ;pop      ebp
3236 00005763 C3         <1>      retn
3237                    <1>
3238                    <1> deallocate_user_pages:
3239                    <1>      ; 20/02/2017
3240                    <1>      ; 19/02/2017 (TRDOS 386 = TRDOS v2.0)
3241                    <1>      ;
3242                    <1>      ; Deallocate virtually contiguous user pages (memory block)
3243                    <1>      ; (caller: 'sysdalloc' system call)
3244                    <1>      ;
3245                    <1>      ; INPUT ->
3246                    <1>      ;      EBX = VIRTUAL ADDRESS (beginning address)
3247                    <1>      ;      ECX = byte count
3248                    <1>      ;      [u.pgdir] = user's page directory
3249                    <1>      ;      [u.ppdire] = parent's page directory
3250                    <1>      ;
3251                    <1>      ; OUTPUT ->
3252                    <1>      ;      If CF = 0
3253                    <1>      ;      EAX = Deallocated memory bytes
3254                    <1>      ;      (Even if shared or read only pages will not be
3255                    <1>      ;      deallocated on M.A.T., this byte count will be
3256                    <1>      ;      returned as virtually deallocated bytes; in fact
3257                    <1>      ;      virtually deallocated user pages * 4096.)
3258                    <1>      ;      EBX = Virtual address (as rounded up)
3259                    <1>      ;      If CF = 1
3260                    <1>      ;      EAX = 0 (there is not any deallocated pages)
3261                    <1>      ;
3262                    <1>      ; Note: Empty page tables will not be deallocated!!!
3263                    <1>      ;      (they will be deallocated at process termination stage)
3264                    <1>      ;
3265                    <1>      ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP
3266                    <1>      ;
3267 00005764 89DE       <1>      mov      esi, ebx
3268 00005766 89F7       <1>      mov      edi, esi
3269 00005768 01CF       <1>      add      edi, ecx
3270 0000576A 81C6FF0F0000 <1>      add      esi, PAGE_SIZE - 1 ; 4095 (round up)
3271 00005770 C1EE0C     <1>      shr      esi, PAGE_SHIFT
3272 00005773 C1EF0C     <1>      shr      edi, PAGE_SHIFT
3273 00005776 89F8       <1>      mov      eax, edi ; end page
3274 00005778 29F0       <1>      sub      eax, esi ; end page - start page
3275 0000577A 0F86D6000000 <1>      jna      da_u_pd_err ; < 1
3276 00005780 89F3       <1>      mov      ebx, esi
3277 00005782 C1E30C     <1>      shl      ebx, PAGE_SHIFT ; virtual address (as rounded up)
3278 00005785 53         <1>      push     ebx ; *
3279 00005786 89C1       <1>      mov      ecx, eax ; page count
3280 00005788 C1E00C     <1>      shl      eax, PAGE_SHIFT ; byte count as adjusted
3281 0000578B 50         <1>      push     eax ; **
3282 0000578C 8B1D[004F0100] <1>      mov      ebx, [u.pgdir] ; physical addr of user's page dir
3283 00005792 81C600040000 <1>      add      esi, CORE/PAGE_SIZE
3284 00005798 89F7       <1>      mov      edi, esi
3285 0000579A 57         <1>      push     edi ; *** ; PTE index (of page directory)
3286 0000579B C1EE0A     <1>      shr      esi, PAGE_D_SHIFT - PAGE_SHIFT ; 22-12=10
3287 0000579E 89F2       <1>      mov      edx, esi
3288                    <1>      ; EDX = PDE index
3289 000057A0 C1E602     <1>      shl      esi, 2 ; convert PDE index to dword offset
3290 000057A3 01DE       <1>      add      esi, ebx ; add page directory address
3291                    <1> da_u_pd_1:
3292 000057A5 AD         <1>      lodsd
3293                    <1>      ;
3294 000057A6 89F5       <1>      mov      ebp, esi ; 20/02/2017
3295                    <1>      ; EBP = next PDE address
3296                    <1>      ;
3297 000057A8 A801       <1>      test     al, PDE_A_PRESENT ; bit 0, present flag (must be 1)

```

```

3298 000057AA 0F8493000000    <1>      jz      da_u_pd_2
3299 000057B0 662500F0        <1>      and     ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3300                                <1>      ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE
3301 000057B4 8B3C24          <1>      mov     edi, [esp] ; ***
3302                                <1>      ; EDI = PTE index (of complete page directory)
3303 000057B7 81E7FF030000    <1>      and     edi, PTE_MASK ; PTE entry in the page table
3304 000057BD C1E702          <1>      shl     edi, 2 ; convert PTE index to dword offset
3305 000057C0 89FE          <1>      mov     esi, edi ; PTE offset in page table (0-4092)
3306 000057C2 01C6          <1>      add     esi, eax ; now, esi points to requested PTE
3307                                <1> da_u_pt_0:
3308 000057C4 AD              <1>      lodsd
3309 000057C5 A801          <1>      test    al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
3310 000057C7 743F          <1>      jz      short da_u_pt_1
3311                                <1>      ;
3312 000057C9 A802          <1>      test    al, PTE_A_WRITE ; bit 1, writable (r/w) flag
3313                                <1>      ; (must be 1)
3314 000057CB 7549          <1>      jnz     short da_u_pt_3
3315                                <1>      ; Read only -duplicated- page (belongs to a parent or a child)
3316 000057CD 66A90002    <1>      test    ax, PTE_DUPLICATED ; Was this page duplicated
3317                                <1>      ; as child's page ?
3318 000057D1 744E          <1>      jz      short da_u_pt_4 ; Clear PTE but don't deallocate the page!
3319                                <1>      ;
3320                                <1>      ; check the parent's PTE value is read only & same page or not..
3321                                <1>      ; EDX = page directory entry index (0-1023)
3322 000057D3 52              <1>      push    edx ; ****
3323                                <1>      ; EDI = page table entry offset (0-4092)
3324 000057D4 8B1D[044F0100] <1>      mov     ebx, [u.ppgdir] ; page directory of the parent process
3325 000057DA 66C1E202    <1>      shl     dx, 2 ; *4
3326 000057DE 01D3          <1>      add     ebx, edx ; PTE address,0 (for the parent)
3327 000057E0 8B13          <1>      mov     edx, [ebx] ; page table address
3328 000057E2 F6C201          <1>      test    dl, PDE_A_PRESENT ; present (valid) or not ?
3329 000057E5 742E          <1>      jz      short da_u_pt_2 ; parent process does not use this page
3330 000057E7 6681E200F0    <1>      and     dx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3331                                <1>      ; EDI = page table entry offset (0-4092)
3332 000057EC 01D7          <1>      add     edi, edx ; PTE address (for the parent)
3333 000057EE 8B1F          <1>      mov     ebx, [edi]
3334 000057F0 F6C301          <1>      test    bl, PTE_A_PRESENT ; present or not ?
3335 000057F3 7420          <1>      jz      short da_u_pt_2 ; parent process does not use this page
3336 000057F5 662500F0    <1>      and     ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3337 000057F9 6681E300F0    <1>      and     bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3338 000057FE 39D8          <1>      cmp     eax, ebx ; parent's and child's pages are same ?
3339 00005800 7513          <1>      jne     short da_u_pt_2 ; not same page
3340                                <1>      ; deallocate the child's page
3341 00005802 800F02          <1>      or      byte [edi], PTE_A_WRITE ; convert to writable page (parent)
3342 00005805 5A              <1>      pop     edx ; ****
3343 00005806 EB19          <1>      jmp     short da_u_pt_4
3344                                <1> da_u_pt_1:
3345 00005808 09C0          <1>      or      eax, eax ; swapped page ?
3346 0000580A 741C          <1>      jz      short da_u_pt_5 ; no
3347                                <1>      ; yes
3348 0000580C D1E8          <1>      shr     eax, 1
3349 0000580E E8EBF9FFFF    <1>      call    unlink_swap_block ; Deallocate swapped page block
3350                                <1>      ; on the swap disk (or in file)
3351 00005813 EB13          <1>      jmp     short da_u_pt_5
3352                                <1> da_u_pt_2:
3353 00005815 5A              <1>      pop     edx ; ****
3354                                <1> da_u_pt_3:
3355 00005816 66A90004    <1>      test    ax, PTE_SHARED ; shared or direct memory access indicator
3356 0000581A 7505          <1>      jnz     short da_u_pt_4 ; AVL bit 1 = 1, do not deallocate this page!
3357                                <1>      ;
3358                                <1>      ;and ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3359 0000581C E836F5FFFF    <1>      call    deallocate_page ; set the mem allocation bit of this page
3360                                <1> da_u_pt_4:
3361 00005821 C746FC00000000 <1>      mov     dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
3362                                <1> da_u_pt_5:
3363 00005828 58              <1>      pop     eax ; *** PTE index (of page directory)
3364 00005829 49              <1>      dec     ecx ; remain page count
3365 0000582A 7427          <1>      jz      short da_u_pd_3
3366 0000582C 40              <1>      inc     eax ; next PTE
3367 0000582D 50              <1>      push    eax ; *** ; save again
3368 0000582E 89C7          <1>      mov     edi, eax
3369 00005830 81E7FF030000    <1>      and     edi, PTE_MASK ; PTE entry index in the page table
3370 00005836 81FF00040000    <1>      cmp     edi, PAGE_SIZE / 4 ; 1024
3371 0000583C 7305          <1>      jnb     short da_u_pd_2
3372 0000583E C1E702          <1>      shl     edi, 2 ; convert index to dword offset
3373 00005841 EB81          <1>      jmp     short da_u_pt_0
3374                                <1> da_u_pd_2:
3375 00005843 42              <1>      inc     edx
3376 00005844 81FA00040000    <1>      cmp     edx, 1024
3377 0000584A 7307          <1>      jnb     short da_u_pd_3
3378 0000584C 89EE          <1>      mov     esi, ebp ; 20/02/2017
3379 0000584E E952FFFFFFF    <1>      jmp     da_u_pd_1
3380                                <1> da_u_pd_3:
3381 00005853 58              <1>      pop     eax ; **
3382 00005854 5B              <1>      pop     ebx ; *
3383 00005855 C3              <1>      retn
3384                                <1>
3385                                <1> da_u_pd_err:
3386 00005856 31C0          <1>      xor     eax, eax
3387 00005858 F9              <1>      stc
3388 00005859 C3              <1>      retn
3389                                <1>
3390                                <1> allocate_user_pages:
3391                                <1>      ; 04/03/2017
3392                                <1>      ; 20/02/2017 (TRDOS 386 = TRDOS v2.0)
3393                                <1>      ;
3394                                <1>      ; Allocate physically contiguous user pages (memory block)
3395                                <1>      ; (caller: 'sysalloc' system call)
3396                                <1>      ;
3397                                <1>      ; INPUT ->
3398                                <1>      ; EAX = PHYSICAL ADDRESS (beginning address)
3399                                <1>      ; EBX = VIRTUAL ADDRESS (beginning address)

```



```

3400      <1>      ;      ECX = byte count
3401      <1>      ;      [u.pgdir] = user's page directory
3402      <1>      ;
3403      <1>      ;      Note: All addresses are (must be) already adjusted
3404      <1>      ;      to page      borders, otherwise, lower 12bits of addresses
3405      <1>      ;      and byte count would be truncated.
3406      <1>      ;
3407      <1>      ;      OUTPUT ->
3408      <1>      ;      none
3409      <1>      ;
3410      <1>      ;      CF = 1 -> insufficient memory error
3411      <1>      ;
3412      <1>      ;      Note: All pages will be allocated in physical page order
3413      <1>      ;      from the beginning page address.
3414      <1>      ;      * A new page table will be added to the page dir
3415      <1>      ;      when the requested PDE is invalid.
3416      <1>      ;      * Those pages will not be added to swap queue
3417      <1>      ;      because main purpose of this allocation is to
3418      <1>      ;      set a direct memory access (DMA controller) buffer.
3419      <1>      ;      (Swapping out a page in a DMA buffer would be wrong!)
3420      <1>      ;      * Previous content of page tables (PTEs) would be
3421      <1>      ;      (should be) deallocated before entering this
3422      <1>      ;      procedure. So, new page table entries (PTEs)
3423      <1>      ;      directly will be written without checking
3424      <1>      ;      their previous content.
3425      <1>      ;      * Only solution to increase free memory by removing
3426      <1>      ;      that non-swappable memory block is to terminate
3427      <1>      ;      the process or to wait until the process will
3428      <1>      ;      deallocate that memory block as itself. ('sysdalloc')
3429      <1>      ;      (No problem, if the process does not grab all of
3430      <1>      ;      -very big amount of- free memory by using
3431      <1>      ;      'sysalloc' system call!?)
3432      <1>      ;      (Even if the process has grabbed all of free memory,
3433      <1>      ;      no problem if the process is not running in
3434      <1>      ;      multitasking mode. No problem in multitasking
3435      <1>      ;      mode if there is not another process which is running
3436      <1>      ;      or waiting or sleeping for an event as it's pages
3437      <1>      ;      are swapped-out. But a new process can not start to
3438      <1>      ;      run if all of free memory has been allocated
3439      <1>      ;      by running processes. Deallocation -'sysdalloc'-
3440      <1>      ;      or terminate a running process is needed
3441      <1>      ;      in order to run a new process.)
3442      <1>      ;
3443      <1>      ;      Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP
3444      <1>      ;
3445      0000585A 89C5      <1>      mov     ebp, eax ; physical address
3446      0000585C 89DE      <1>      mov     esi, ebx
3447      0000585E 81C600004000 <1>      add     esi, CORE ; start of user's memory (4M)
3448      00005864 C1EE0C      <1>      shr     esi, PAGE_SHIFT
3449      00005867 C1E90C      <1>      shr     ecx, PAGE_SHIFT
3450      0000586A 8B1D[004F0100] <1>      mov     ebx, [u.pgdir] ; physical addr of user's page dir
3451      00005870 89F7      <1>      mov     edi, esi
3452      00005872 57      <1>      push    edi ; * ; PTE index (in page directory)
3453      00005873 C1EE0A      <1>      shr     esi, PAGE_D_SHIFT - PAGE_SHIFT ; 22-12=10
3454      00005876 89F2      <1>      mov     edx, esi
3455      <1>      ;      EDX = PDE index
3456      00005878 C1E602      <1>      shl     esi, 2 ; convert PDE index to dword offset
3457      0000587B 01DE      <1>      add     esi, ebx ; add page directory address
3458      <1>      a_u_pd_0:
3459      0000587D AD      <1>      lodsd
3460      <1>      ;
3461      0000587E 89F3      <1>      mov     ebx, esi ; next PDE address
3462      <1>      ;
3463      00005880 A801      <1>      test    al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
3464      00005882 7515      <1>      jnz     short a_u_pd_2
3465      <1>      ;
3466      <1>      ;      empty PDE (it does not point to valid page table address)
3467      00005884 E8F0F2FFFF      <1>      call    allocate_page ; (allocate a new page table)
3468      00005889 7302      <1>      jnc     short a_u_pd_1 ; OK... now, we have a new page table.
3469      <1>      ;      cf = 1
3470      <1>      ;      There is not a free memory page to allocate a new page table !!!
3471      0000588B 5E      <1>      pop     esi ; *
3472      0000588C C3      <1>      retn    ; return to 'sysalloc' with 'insufficient memory' error
3473      <1>      ;
3474      <1>      a_u_pd_1: ; clear the new page table content
3475      <1>      ;      EAX = Physical (base) address of the new page table
3476      0000588D 89C1      <1>      mov     ecx, eax
3477      0000588F E85FF3FFFF      <1>      call    clear_page ; Clear page content
3478      <1>      ;
3479      00005894 0C07      <1>      or      al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
3480      <1>      ;      ; set bit 0, bit 1 and bit 2 to 1
3481      <1>      ;      ; (present, writable, user)
3482      00005896 8946FC      <1>      mov     [esi-4], eax
3483      <1>      a_u_pd_2:
3484      00005899 662500F0      <1>      and     ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3485      <1>      ;      EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE
3486      0000589D 8B3C24      <1>      mov     edi, [esp] ; ***
3487      <1>      ;      EDI = PTE index (of page directory)
3488      000058A0 81E7FF030000      <1>      and     edi, PTE_MASK ; PTE entry index in the page table
3489      <1>      ;      EBX = next PDE address
3490      000058A6 89FE      <1>      mov     esi, edi ; PTE index in page table (0-1023)
3491      000058A8 C1E702      <1>      shl     edi, 2 ; convert PTE index to dword offset
3492      000058AB 01C7      <1>      add     edi, eax ; now, edi points to requested PTE
3493      <1>      a_u_pt_0:
3494      000058AD 89E8      <1>      mov     eax, ebp ; physical address
3495      000058AF 0C07      <1>      or      al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 04/03/2017
3496      000058B1 AB      <1>      stosd
3497      000058B2 49      <1>      dec     ecx ; remain page count
3498      000058B3 741C      <1>      jz      short a_u_pd_4
3499      000058B5 81C500100000      <1>      add     ebp, PAGE_SIZE
3500      000058BB 46      <1>      inc     esi ; next PTE (index)
3501      000058BC 81FE00040000      <1>      cmp     esi, PAGE_SIZE/4 ; 1024

```

```

3502 000058C2 72E9      <1>      jb      short a_u_pt_0
3503                    <1> a_u_pd_3:
3504 000058C4 42          <1>      inc      edx
3505 000058C5 81FA00040000 <1>      cmp      edx, 1024
3506 000058CB 7304      <1>      jnb      short a_u_pd_4
3507 000058CD 89DE      <1>      mov      esi, ebx ; the next PDE address
3508 000058CF EBAC      <1>      jmp      short a_u_pd_0
3509                    <1> a_u_pd_4:
3510 000058D1 5F          <1>      pop      edi ; *
3511 000058D2 C3          <1>      retn
3512                    <1>
3513                    <1> ; /// End Of MEMORY MANAGEMENT FUNCTIONS ///
3514                    <1>
3515                    <1> ;; Data:
3516                    <1>
3517                    <1> ; 09/03/2015
3518                    <1> ;swpq_count: dw 0 ; count of pages on the swap que
3519                    <1> ;swp_drv:  dd 0 ; logical drive description table address of the swap drive/disk
3520                    <1> ;swpd_size: dd 0 ; size of swap drive/disk (volume) in sectors (512 bytes).

3521                    <1> ;swpd_free: dd 0 ; free page blocks (4096 bytes) on swap disk/drive (logical)
3522                    <1> ;swpd_next: dd 0 ; next free page block
3523                    <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 000058D3 9C          <1>      pushfd
30 000058D4 0E          <1>      push  cs
31 000058D5 E801000000 <1>      call  TIME_OF_DAY_1
32 000058DA 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     000058DB 80642408FE <1>     and     byte [esp+8], 11111110b ; clear carry bit of eflags register
113     <1>     ;
114     000058E0 80FC08 <1>     cmp     ah, (RTC_TBE-RTC_TB)/4 ; CHECK IF COMMAND IN VALID RANGE (0-7)
115     000058E3 F5 <1>     cmc     ; COMPLEMENT CARRY FOR ERROR EXIT
116     <1>     ; (*) jc short TIME_9 ; EXIT WITH CARRY = 1 IF NOT VALID
117     000058E4 721A <1>     jc      short _TIME_9 ; 29/05/2016
118     <1>
119     000058E6 1E <1>     push    ds
120     000058E7 56 <1>     push    esi
121     000058E8 66BE1000 <1>     mov     si, KDATA ; kernel data segment
122     000058EC 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     000058EE C0E402 <1>     shl     ah, 2 ; convert function to dword offset
131     000058F1 0FB6F4 <1>     movzx   esi, ah ; PLACE INTO ADDRESSING REGISTER
132     <1>     ;cli ; NO INTERRUPTS DURING TIME FUNCTIONS
133     000058F4 FF96[06590000] <1>     call    [esi+RTC_TB] ; VECTOR TO FUNCTION REQUESTED WITH CY=0
134     <1>     ; RETURN WITH CARRY FLAG SET FOR RESULT
135     <1>     ; INTERRUPTS BACK ON
136     000058FA B400 <1>     mov     ah, 0 ; CLEAR (AH) TO ZERO
137     000058FC 5E <1>     pop     esi ; RECOVER USERS REGISTER
138     000058FD 1F <1>     pop     ds ; RECOVER USERS SEGMENT SELECTOR
139     <1>
140     <1>     ;;15/01/2017
141     <1>     ; 02/01/2017
142     <1>     ;mov byte [ss:intflg], 0 ; 07/01/2017
143     <1>
144     <1> ;TIME_9:
145     <1>     ; RETURN WITH CY= 0 IF NO ERROR
146     <1>     ; (*) 29/05/2016
147     <1>     ; (*) retf 4 ; skip eflags on stack
148     000058FE 7305 <1>     jnc     short _TIME_10
149     <1> _TIME_9:
150     <1>     ; 29/05/2016 -set carry flag on stack-
151     <1>     ; [esp] = EIP
152     <1>     ; [esp+4] = CS
153     <1>     ; [esp+8] = E-FLAGS
154     00005900 804C240801 <1>     or      byte [esp+8], 1 ; set carry bit of eflags register
155     <1>     ; [esp+12] = ESP (user)
156     <1>     ; [esp+16] = SS (User)
157     <1> _TIME_10:
158     00005905 CF <1>     iretd
159     <1>
160     <1>     ; (*) 29/05/2016 - 'ref 4' intruction causes to stack fault
161     <1>     ; (OUTER-PRIVILEGE-LEVEL)
162     <1>     ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986
163     <1>     ; // RETF instruction:
164     <1>     ;
165     <1>     ; IF OperandMode=32 THEN
166     <1>     ; Load CS:EIP from stack;
167     <1>     ; Set CS RPL to CPL;
168     <1>     ; Increment eSP by 8 plus the immediate offset if it exists;
169     <1>     ; Load SS:eSP from stack;
170     <1>     ; ELSE (* OperandMode=16 *)
171     <1>     ; Load CS:IP from stack;
172     <1>     ; Set CS RPL to CPL;
173     <1>     ; Increment eSP by 4 plus the immediate offset if it exists;
174     <1>     ; Load SS:eSP from stack;
175     <1>     ; FI;
176     <1>     ;
177     <1>     ; //
178     <1>     ; ROUTINE VECTOR TABLE (AH)=
179     <1> RTC_TB:
180     00005906 [26590000] <1>     dd      RTC_00 ; 0 = READ CURRENT CLOCK COUNT

```

```

181 0000590A [39590000] <1> dd RTC_10 ; 1 = SET CLOCK COUNT
182 0000590E [47590000] <1> dd RTC_20 ; 2 = READ THE REAL TIME CLOCK TIME
183 00005912 [76590000] <1> dd RTC_30 ; 3 = SET REAL TIME CLOCK TIME
184 00005916 [B8590000] <1> dd RTC_40 ; 4 = READ THE REAL TIME CLOCK DATE
185 0000591A [E5590000] <1> dd RTC_50 ; 5 = SET REAL TIME CLOCK DATE
186 0000591E [325A0000] <1> dd RTC_60 ; 6 = SET THE REAL TIME CLOCK ALARM
187 00005922 [855A0000] <1> dd RTC_70 ; 7 = RESET ALARM
188 <1>
189 <1> RTC_TBE equ $
190 <1>
191 <1> RTC_00: ; READ TIME COUNT
192 00005926 A0[A4380100] <1> mov al, [TIMER_OFL] ; GET THE OVERFLOW FLAG
193 0000592B C605[A4380100]00 <1> mov byte [TIMER_OFL], 0 ; AND THEN RESET THE OVERFLOW FLAG
194 00005932 8B0D[A0380100] <1> mov ecx, [TIMER_LH] ; GET COUNT OF TIME
195 00005938 C3 <1> retn
196 <1>
197 <1> RTC_10: ; SET TIME COUNT
198 00005939 890D[A0380100] <1> mov [TIMER_LH], ecx ; SET TIME COUNT
199 0000593F C605[A4380100]00 <1> mov byte [TIMER_OFL], 0 ; RESET OVERFLOW FLAG
200 00005946 C3 <1> retn ; RETURN WITH NO CARRY
201 <1>
202 <1> RTC_20: ; GET RTC TIME
203 00005947 E8EB010000 <1> call UPD_IPR ; CHECK FOR UPDATE IN PROCESS
204 0000594C 7227 <1> jc short RTC_29 ; EXIT IF ERROR (CY= 1)
205 <1>
206 0000594E B000 <1> mov al, CMOS_SECONDS ; SET ADDRESS OF SECONDS
207 00005950 E8FD010000 <1> call CMOS_READ ; GET SECONDS
208 00005955 88C6 <1> mov dh, al ; SAVE
209 00005957 B00B <1> mov al, CMOS_REG_B ; ADDRESS ALARM REGISTER
210 00005959 E8F4010000 <1> call CMOS_READ ; READ CURRENT VALUE OF DSE BIT
211 0000595E 2401 <1> and al, 00000001b ; MASK FOR VALID DSE BIT
212 00005960 88C2 <1> mov dl, al ; SET [DL] TO ZERO FOR NO DSE BIT
213 00005962 B002 <1> mov al, CMOS_MINUTES ; SET ADDRESS OF MINUTES
214 00005964 E8E9010000 <1> call CMOS_READ ; GET MINUTES
215 00005969 88C1 <1> mov cl, al ; SAVE
216 0000596B B004 <1> mov al, CMOS_HOURS ; SET ADDRESS OF HOURS
217 0000596D E8E0010000 <1> call CMOS_READ ; GET HOURS
218 00005972 88C5 <1> mov ch, al ; SAVE
219 00005974 F8 <1> clc ; SET CY= 0
220 <1> RTC_29:
221 00005975 C3 <1> retn ; RETURN WITH RESULT IN CARRY FLAG
222 <1>
223 <1> RTC_30: ; SET RTC TIME
224 00005976 E8BC010000 <1> call UPD_IPR ; CHECK FOR UPDATE IN PROCESS
225 0000597B 7305 <1> jnc short RTC_35 ; GO AROUND IF CLOCK OPERATING
226 0000597D E817010000 <1> call RTC_STA ; ELSE TRY INITIALIZING CLOCK
227 <1> RTC_35:
228 00005982 88F4 <1> mov ah, dh ; GET TIME BYTE - SECONDS
229 00005984 B000 <1> mov al, CMOS_SECONDS ; ADDRESS SECONDS
230 00005986 E8E0010000 <1> call CMOS_WRITE ; UPDATE SECONDS
231 0000598B 88CC <1> mov ah, cl ; GET TIME BYTE - MINUTES
232 0000598D B002 <1> mov al, CMOS_MINUTES ; ADDRESS MINUTES
233 0000598F E8D7010000 <1> call CMOS_WRITE ; UPDATE MINUTES
234 00005994 88EC <1> mov ah, ch ; GET TIME BYTE - HOURS
235 00005996 B004 <1> mov al, CMOS_HOURS ; ADDRESS HOURS
236 00005998 E8CE010000 <1> call CMOS_WRITE ; UPDATE ADDRESS
237 <1> ;mov al, CMOS_REG_B ; ADDRESS ALARM REGISTER
238 <1> ;mov ah, al
239 0000599D 66B80B0B <1> mov ax, CMOS_REG_B * 257
240 000059A1 E8AC010000 <1> call CMOS_READ ; READ CURRENT TIME
241 000059A6 2462 <1> and al, 01100010b ; MASK FOR VALID BIT POSITIONS
242 000059A8 0C02 <1> or al, 00000010b ; TURN ON 24 HOUR MODE
243 000059AA 80E201 <1> and dl, 00000001b ; USE ONLY THE DSE BIT
244 000059AD 08D0 <1> or al, dl ; GET DAY LIGHT SAVINGS TIME BIT (OSE)
245 000059AF 86E0 <1> xchg ah, al ; PLACE IN WORK REGISTER AND GET ADDRESS
246 000059B1 E8B5010000 <1> call CMOS_WRITE ; SET NEW ALARM SITS
247 000059B6 F8 <1> clc ; SET CY= 0
248 000059B7 C3 <1> retn ; RETURN WITH CY= 0
249 <1>
250 <1> RTC_40: ; GET RTC DATE
251 000059B8 E87A010000 <1> call UPD_IPR ; CHECK FOR UPDATE IN PROCESS
252 000059BD 7225 <1> jc short RTC_49 ; EXIT IF ERROR (CY= 1)
253 <1>
254 000059BF B007 <1> mov al, CMOS_DAY_MONTH ; ADDRESS DAY OF MONTH
255 000059C1 E88C010000 <1> call CMOS_READ ; READ DAY OF MONTH
256 000059C6 88C2 <1> mov dl, al ; SAVE
257 000059C8 B008 <1> mov al, CMOS_MONTH ; ADDRESS MONTH
258 000059CA E883010000 <1> call CMOS_READ ; READ MONTH
259 000059CF 88C6 <1> mov dh, al ; SAVE
260 000059D1 B009 <1> mov al, CMOS_YEAR ; ADDRESS YEAR
261 000059D3 E87A010000 <1> call CMOS_READ ; READ YEAR
262 000059D8 88C1 <1> mov cl, al ; SAVE
263 000059DA B032 <1> mov al, CMOS_CENTURY ; ADDRESS CENTURY LOCATION
264 000059DC E871010000 <1> call CMOS_READ ; GET CENTURY BYTE
265 000059E1 88C5 <1> mov ch, al ; SAVE
266 000059E3 F8 <1> clc ; SET CY=0
267 <1> RTC_49:
268 000059E4 C3 <1> retn ; RETURN WITH RESULTS IN CARRY FLAG
269 <1>
270 <1> RTC_50: ; SET RTC DATE
271 000059E5 E84D010000 <1> call UPD_IPR ; CHECK FOR UPDATE IN PROCESS
272 000059EA 7305 <1> jnc short RTC_55 ; GO AROUND IF NO ERROR
273 000059EC E8A8000000 <1> call RTC_STA ; ELSE INITIALIZE CLOCK
274 <1> RTC_55:
275 000059F1 66B80600 <1> mov ax, CMOS_DAY_WEEK ; ADDRESS OF DAY OF WEEK BYTE
276 000059F5 E871010000 <1> call CMOS_WRITE ; LOAD ZEROS TO DAY OF WEEK
277 000059FA 88D4 <1> mov ah, dl ; GET DAY OF MONTH BYTE
278 000059FC B007 <1> mov al, CMOS_DAY_MONTH ; ADDRESS DAY OF MONTH BYTE
279 000059FE E868010000 <1> call CMOS_WRITE ; WRITE OF DAY OF MONTH REGISTER
280 00005A03 88F4 <1> mov ah, dh ; GET MONTH
281 00005A05 B008 <1> mov al, CMOS_MONTH ; ADDRESS MONTH BYTE
282 00005A07 E85F010000 <1> call CMOS_WRITE ; WRITE MONTH REGISTER

```



```

283 00005A0C 88CC      <1>      mov     ah, cl          ; GET YEAR BYTE
284 00005A0E B009      <1>      mov     al, CMOS_YEAR      ; ADDRESS YEAR REGISTER
285 00005A10 E856010000 <1>      call    CMOS_WRITE      ; WRITE YEAR REGISTER
286 00005A15 88EC      <1>      mov     ah, ch          ; GET CENTURY BYTE
287 00005A17 B032      <1>      mov     al, CMOS_CENTURY    ; ADDRESS CENTURY BYTE
288 00005A19 E84D010000 <1>      call    CMOS_WRITE      ; WRITE CENTURY LOCATION
289                      <1>      ;mov     al, CMOS_REG_B      ; ADDRESS ALARM REGISTER
290                      <1>      ;mov     ah, al
291 00005A1E 66B80B0B   <1>      mov     ax, CMOS_REG_B * 257
292 00005A22 E82B010000 <1>      call    CMOS_READ      ; READ WIRRENT SETTINGS
293 00005A27 247F      <1>      and     al, 07Fh        ; CLEAR 'SET BIT'
294 00005A29 86E0      <1>      xchg    ah, al          ; MOVE TO WORK REGISTER
295 00005A2B E83B010000 <1>      call    CMOS_WRITE      ; AND START CLOCK UPDATING
296 00005A30 F8        <1>      cld          ; SET CY= 0
297 00005A31 C3        <1>      retn         ; RETURN CY=0
298                      <1>
299                      <1> RTC_60:          ; SET RTC ALARM
300 00005A32 B00B      <1>      mov     al, CMOS_REG_B      ; ADDRESS ALARM
301 00005A34 E819010000 <1>      call    CMOS_READ      ; READ ALARM REGISTER
302 00005A39 A820      <1>      test    al, 20h        ; CHECK FOR ALARM ALREADY ENABLED
303 00005A3B F9        <1>      stc          ; SET CARRY IN CASE OF ERROR
304 00005A3C 7542      <1>      jnz     short RTC_69    ; ERROR EXIT IF ALARM SET
305 00005A3E E8F4000000 <1>      call    UPD_IPR        ; CHECK FOR UPDATE IN PROCESS
306 00005A43 7305      <1>      jnc     short RTC_65    ; SKIP INITIALIZATION IF NO ERROR
307 00005A45 E84F000000 <1>      call    RTC_STA        ; ELSE INITIALIZE CLOCK
308                      <1> RTC_65:
309 00005A4A 88F4      <1>      mov     ah, dh          ; GET SECONDS BYTE
310 00005A4C B001      <1>      mov     al, CMOS_SEC_ALARM    ; ADDRESS THE SECONDS ALARM REGISTER
311 00005A4E E818010000 <1>      call    CMOS_WRITE      ; INSERT SECONDS
312 00005A53 88CC      <1>      mov     ah, cl          ; GET MINUTES PARAMETER
313 00005A55 B003      <1>      mov     al, CMOS_MIN_ALARM    ; ADDRESS MINUTES ALARM REGISTER
314 00005A57 E80F010000 <1>      call    CMOS_WRITE      ; INSERT MINUTES
315 00005A5C 88EC      <1>      mov     ah, ch          ; GET HOURS PARAMETER
316 00005A5E B005      <1>      mov     al, CMOS_HR_ALARM    ; ADDRESS HOUR ALARM REGISTER
317 00005A60 E806010000 <1>      call    CMOS_WRITE      ; INSERT HOURS
318 00005A65 E4A1      <1>      in      al, INTB01      ; READ SECOND INTERRUPT MASK REGISTER
319 00005A67 24FE      <1>      and     al, 0FEh        ; ENABLE ALARM TIMER BIT (CY= 0)
320 00005A69 E6A1      <1>      out     INTB01, al      ; WRITE UPDATED MASK
321                      <1>      ;mov     al, CMOS_REG_B      ; ADDRESS ALARM REGISTER
322                      <1>      ;mov     ah, al
323 00005A6B 66B80B0B   <1>      mov     ax, CMOS_REG_B * 257
324 00005A6F E8DE000000 <1>      call    CMOS_READ      ; READ CURRENT ALARM REGISTER
325 00005A74 247F      <1>      and     al, 07Fh        ; ENSURE SET BIT TURNED OFF
326 00005A76 0C20      <1>      or      al, 20h        ; TURN ON ALARM ENABLE
327 00005A78 86E0      <1>      xchg    ah, al          ; MOVE MASK TO OUTPUT REGISTER
328 00005A7A E8EC000000 <1>      call    CMOS_WRITE      ; WRITE NEW ALARM MASK
329 00005A7F F8        <1>      cld          ; SET CY= 0
330                      <1> RTC_69:
331 00005A80 66B80000   <1>      mov     ax, 0          ; CLEAR AX REGISTER
332 00005A84 C3        <1>      retn         ; RETURN WITH RESULTS IN CARRY FLAG
333                      <1>
334                      <1> RTC_70:          ; RESET ALARM
335                      <1>      ;mov     al, CMOS_REG_B      ; ADDRESS ALARM REGISTER
336                      <1>      ;mov     ah, al
337 00005A85 66B80B0B   <1>      mov     ax, CMOS_REG_B * 257    ; ADDRESS ALARM REGISTER (TO BOTH AH,AL)
338 00005A89 E8C4000000 <1>      call    CMOS_READ      ; READ ALARM REGISTER
339 00005A8E 2457      <1>      and     al, 57h        ; TURN OFF ALARM ENABLE
340 00005A90 86E0      <1>      xchg    ah, al          ; SAVE DATA AND RECOVER ADDRESS
341 00005A92 E8D4000000 <1>      call    CMOS_WRITE      ; RESTORE NEW VALUE
342 00005A97 F8        <1>      cld          ; SET CY= 0
343 00005A98 C3        <1>      retn         ; RETURN WITH NO CARRY
344                      <1>
345                      <1> RTC_STA:          ; INITIALIZE REAL TIME CLOCK
346                      <1>      ;mov     al, CMOS_REG_A      ; ADDRESS REGISTER A AND LOAD DATA MASK
347                      <1>      ;mov     ah, 26h
348 00005A99 66B80A26   <1>      mov     ax, (26h*100h)+CMOS_REG_A
349 00005A9D E8C9000000 <1>      call    CMOS_WRITE      ; INITIALIZE STATUS REGISTER A
350                      <1>      ;mov     al, CMOS_REG_B      ; SET "SET BIT" FOR CLOCK INITIALIZATION
351                      <1>      ;mov     ah, 82h
352 00005AA2 66B80B82   <1>      mov     ax, (82h*100h)+CMOS_REG_B
353 00005AA6 E8C0000000 <1>      call    CMOS_WRITE      ; AND 24 HOUR MODE TO REGISTER B
354 00005AAB B00C      <1>      mov     al, CMOS_REG_C      ; ADDRESS REGISTER C
355 00005AAD E8A0000000 <1>      call    CMOS_READ      ; READ REGISTER C TO INITIALIZE
356 00005AB2 B00D      <1>      mov     al, CMOS_REG_D      ; ADDRESS REGISTER D
357 00005AB4 E899000000 <1>      call    CMOS_READ      ; READ REGISTER D TO INITIALIZE
358 00005AB9 C3        <1>      retn
359                      <1>
360                      <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
361                      <1>
362                      <1> ;--- HARDWARE INT 70 H -- ( IRQ LEVEL 8) -----
363                      <1> ; ALARM INTERRUPT HANDLER (RTC) :
364                      <1> ; THIS ROUTINE HANDLES THE PERIODIC AND ALARM INTERRUPTS FROM THE CMOS :
365                      <1> ; TIMER. INPUT FREQUENCY IS 1.024 KHZ OR APPROXIMATELY 1024 INTERRUPTS :
366                      <1> ; EVERY SECOND FOR THE PERIODIC INTERRUPT. FOR THE ALARM FUNCTION, :
367                      <1> ; THE INTERRUPT WILL OCCUR AT THE DESIGNATED TIME. :
368                      <1> ; :
369                      <1> ; INTERRUPTS ARE ENABLED WHEN THE EVENT OR ALARM FUNCTION IS ACTIVATED. :
370                      <1> ; FOR THE EVENT INTERRUPT, THE HANDLER WILL DECREMENT THE WAIT COUNTER :
371                      <1> ; AND WHEN IT EXPIRES WILL SET THE DESIGNATED LOCATION TO 80H. FOR :
372                      <1> ; THE ALARM INTERRUPT. THE USER MUST PROVIDE A ROUTINE TO INTERCEPT :
373                      <1> ; THE CORRECT ADDRESS FROM THE VECTOR TABLE INVOKED BY INTERRUPT 4AH :
374                      <1> ; PRIOR TO SETTING THE REAL TIME CLOCK ALARM (INT 1AH, AH= 06H). :
375                      <1> ;-----
376                      <1>
377                      <1> RTC_A_INT: ; 07/01/2017
378                      <1> ;RTC_INT:          ; ALARM INTERRUPT
379 00005ABA 1E        <1>      push    ds          ; LEAVE INTERRUPTS DISABLED
380 00005ABB 50        <1>      push    eax         ; SAVE REGISTERS
381 00005ABC 57        <1>      push    edi
382                      <1> RTC_I_1:          ; CHECK FOR SECOND INTERRUPT
383 00005ABD 66B88C8B   <1>      mov     ax, 256*(CMOS_REG_B+NMI)+CMOS_REG_C+NMI ; ALARM AND STATUS
384 00005AC1 E670      <1>      out     CMOS_PORT, al    ; WRITE ALARM FLAG MASK ADDRESS

```

```

385 00005AC3 90          <1>      nop                    ; I/O DELAY
386 00005AC4 EB00        <1>      jmp     short $+2
387 00005AC6 E471        <1>      in      al, CMOS_DATA      ; READ AND RESET INTERRUPT REQUEST FLAGS
388 00005AC8 A860        <1>      test    al, 01100000b      ; CHECK FOR EITHER INTERRUPT PENDING
389 00005ACA 745D        <1>      jz      short RTC_I_9      ; EXIT IF NOT A VALID RTC INTERRUPT
390                      <1>
391 00005ACC 86E0        <1>      xchg    ah, al                    ; SAVE FLAGS AND GET ENABLE ADDRESS
392 00005ACE E670        <1>      out     CMOS_PORT, al      ; WRITE ALARM ENABLE MASK ADDRESS
393 00005AD0 90          <1>      nop                    ; I/O DELAY
394 00005AD1 EB00        <1>      jmp     short $+2
395 00005AD3 E471        <1>      in      al, CMOS_DATA      ; READ CURRENT ALARM ENABLE MASK
396 00005AD5 20E0        <1>      and     al, ah                    ; ALLOW ONLY SOURCES THAT ARE ENABLED
397 00005AD7 A840        <1>      test    al, 01000000b      ; CHECK FOR PERIODIC INTERRUPT
398 00005AD9 743B        <1>      jz      short RTC_I_5      ; SKIP IF NOT A PERIODIC INTERRUPT
399                      <1>
400                      <1> ;-----      DECREMENT WAIT COUNT BY INTERRUPT INTERVAL
401                      <1>
402 00005ADB 66BF1000     <1>      mov     di, KDATA          ; kernel data segment
403 00005ADF 8EDF        <1>      mov     ds, di
404                      <1>
405 00005AE1 812D[98380100]D003- <1>      sub     dword [RTC_LH], 976 ; DECREMENT COUNT BY 1/1024
406 00005AE9 0000        <1>
407                      <1>
408                      <1> ;-----      TURN OFF PERIODIC INTERRUPT ENABLE
409                      <1>
410 00005AED 6650        <1>      push    ax                    ; SAVE INTERRUPT FLAG MASK
411 00005AEF 66B88B8B    <1>      mov     ax, 257*(CMOS_REG_B+NMI) ; INTERRUPT ENABLE REGISTER
412 00005AF3 E670        <1>      out     CMOS_PORT, al      ; WRITE ADDRESS TO CMOS CLOCK
413 00005AF5 90          <1>      nop                    ; I/O DELAY
414 00005AF6 EB00        <1>      jmp     short $+2
415 00005AF8 E471        <1>      in      al, CMOS_DATA      ; READ CURRENT ENABLES
416 00005AFA 24BF        <1>      and     al, 0BFh          ; TURN OFF PIE
417 00005AFC 86C4        <1>      xchg    al, ah                    ; GET CMOS ADDRESS AND SAVE VALUE
418 00005AFE E670        <1>      out     CMOS_PORT, al      ; ADDRESS REGISTER B
419 00005B00 86C4        <1>      xchg    al, ah                    ; GET NEW INTERRUPT ENABLE MASK
420 00005B02 E671        <1>      out     CMOS_DATA, al      ; SET MASK IN INTERRUPT ENABLE REGISTER
421 00005B04 C605[9C380100]00 <1>      mov     byte [RTC_WAIT_FLAG], 0 ; SET FUNCTION ACTIVE FLAG OFF
422 00005B0B 8B3D[9D380100] <1>      mov     edi, [USER_FLAG] ; SET UP (DS:DI) TO POINT TO USER FLAG
423 00005B11 C60780      <1>      mov     byte [edi], 80h      ; TURN ON USERS FLAG
424 00005B14 6658        <1>      pop     ax                    ; GET INTERRUPT SOURCE BACK
425                      <1> RTC_I_5:
426 00005B16 A820        <1>      test    al, 00100000b      ; TEST FOR ALARM INTERRUPT
427 00005B18 740D        <1>      jz      short RTC_I_7      ; SKIP USER INTERRUPT CALL IF NOT ALARM
428                      <1>
429 00005B1A B00D        <1>      mov     al, CMOS_REG_D          ; POINT TO DEFAULT READ ONLY REGISTER
430 00005B1C E670        <1>      out     CMOS_PORT, al      ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
431 00005B1E FB          <1>      sti                    ; INTERRUPTS BACK ON NOW
432 00005B1F 52          <1>      push    edx
433 00005B20 E891980000   <1>      call    INT4Ah          ; TRANSFER TO USER ROUTINE
434 00005B25 5A          <1>      pop     edx
435 00005B26 FA          <1>      cli                    ; BLOCK INTERRUPT FOR RETRY
436                      <1> RTC_I_7:
437 00005B27 EB94        <1>      jmp     short RTC_I_1      ; RESTART ROUTINE TO HANDLE DELAYED
438                      <1>
439                      <1> RTC_I_9:
440 00005B29 B00D        <1>      mov     al, CMOS_REG_D          ; EXIT - NO PENDING INTERRUPTS
441 00005B2B E670        <1>      out     CMOS_PORT, al      ; POINT TO DEFAULT READ ONLY REGISTER
442 00005B2D B020        <1>      mov     al, EOI                    ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
443 00005B2F E6A0        <1>      out     INTB00, al          ; END OF INTERRUPT MASK TO 8259 - 2
444 00005B31 E620        <1>      out     INTA00, al          ; TO 8259 - 2
445 00005B33 5F          <1>      pop     edi                    ; TO 8259 - 1
446 00005B34 58          <1>      pop     eax                    ; RESTORE REGISTERS
447 00005B35 1F          <1>      pop     ds
448 00005B36 CF          <1>      iretd                    ; END OF INTERRUPT
449                      <1>
450                      <1>
451                      <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
452                      <1> ; 22/08/2014 (Retro UNIX 386 v1)
453                      <1> ; IBM PC/AT BIOS source code ----- 10/06/85 (bios2.asm)
454                      <1> UPD_IPR:
455 00005B37 51          <1>      push    ecx                    ; WAIT TILL UPDATE NOT IN PROGRESS
456                      <1>
457                      <1> ; 29/05/2016
458 00005B38 B968110000   <1>      mov     ecx, ((1984+244)*4)/2 ; AWARD BIOS 1999, ATIME.ASM
459                      <1> ; 'WAITCPU_CHK_UD_STAT'
460                      <1> ; (244Us + 1984Us)
461                      <1> ; (assume each read takes
462                      <1> ; 2 microseconds).
463                      <1> ;mov     ecx, 65535
464                      <1> ;mov     cx, 800                    ; SET TIMEOUT LOOP COUNT (= 800)
465                      <1> UPD_10:
466 00005B3D B00A        <1>      mov     al, CMOS_REG_A          ; ADDRESS STATUS REGISTER A
467 00005B3F FA          <1>      cli                    ; NO TIMER INTERRUPTS DURING UPDATES
468 00005B40 E80D000000   <1>      call    CMOS_READ          ; READ UPDATE IN PROCESS FLAG
469 00005B45 A880        <1>      test    al, 80h                    ; IF UIP BIT IS ON ( CANNOT READ TIME )
470 00005B47 7406        <1>      jz      short UPD_90      ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
471 00005B49 FB          <1>      sti                    ; ALLOW INTERRUPTS WHILE WAITING
472 00005B4A E2F1        <1>      loop    UPD_10            ; LOOP TILL READY OR TIMEOUT
473 00005B4C 31C0        <1>      xor     eax, eax          ; CLEAR RESULTS IF ERROR
474                      <1> ; xor ax, ax
475 00005B4E F9          <1>      stc                    ; SET CARRY FOR ERROR
476                      <1> UPD_90:
477 00005B4F 59          <1>      pop     ecx                    ; RESTORE CALLERS REGISTER
478 00005B50 FA          <1>      cli                    ; INTERRUPTS OFF DURING SET
479 00005B51 C3          <1>      retn                    ; RETURN WITH CY FLAG SET
480                      <1>
481                      <1>
482                      <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
483                      <1> ; 22/08/2014 (Retro UNIX 386 v1)
484                      <1> ; IBM PC/AT BIOS source code ----- 10/06/85 (test4.asm)
485                      <1>

```

```

486 <1> ;--- CMOS_READ -----
487 <1> ; READ BYTE FROM CMOS_SYSTEM CLOCK CONFIGURATION TABLE :
488 <1> ; :
489 <1> ; INPUT: (AL)= CMOS_TABLE ADDRESS TO BE READ :
490 <1> ; BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT :
491 <1> ; BITS 6-0 = ADDRESS OF TABLE LOCATION TO READ :
492 <1> ; :
493 <1> ; OUTPUT: (AL) VALUE AT LOCATION (AL) MOVED INTO (AL). IF BIT 7 OF (AL) WAS :
494 <1> ; ON THEN NMI LEFT DISABLED, DURING THE CMOS READ BOTH NMI AND :
495 <1> ; NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. :
496 <1> ; THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND :
497 <1> ; THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN. :
498 <1> ; ONLY THE (AL) REGISTER AND THE NMI STATE IS CHANGED. :
499 <1> ;-----
500 <1>
501 <1> CMOS_READ:
502 <1> pushf ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
503 00005B53 D0C0 <1> rol al, 1 ; MOVE NMI BIT TO LOW POSITION
504 00005B55 F9 <1> stc ; FORCE NMI BIT ON IN CARRY FLAG
505 00005B56 D0D8 <1> rcr al, 1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
506 00005B58 FA <1> cli ; DISABLE INTERRUPTS
507 00005B59 E670 <1> out CMOS_PORT, al ; ADDRESS LOCATION AND DISABLE NMI
508 <1> ; 29/05/2016
509 <1> ;nop ; I/O DELAY
510 00005B5B E6EB <1> out 0ebh,al ; NEWIODELAY ; AWARD BIOS 1999, ATIME.ASM
511 <1> ;
512 00005B5D E471 <1> in al, CMOS_DATA ; READ THE REQUESTED CMOS LOCATION
513 00005B5F 6650 <1> push ax ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
514 <1> ; 15/03/2015 ; IBM PC/XT Model 286 BIOS source code
515 <1> ; ----- 10/06/85 (test4.asm)
516 00005B61 B01E <1> mov al, CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
517 <1> ;mov al, CMOS_REG_D*2 ; GET ADDRESS OF DEFAULT LOCATION
518 00005B63 D0D8 <1> rcr al, 1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
519 00005B65 E670 <1> out CMOS_PORT, al ; SET DEFAULT TO READ ONLY REGISTER
520 00005B67 6658 <1> pop ax ; RESTORE (AH) AND (AL), CMOS BYTE
521 00005B69 9D <1> popf
522 00005B6A C3 <1> retn ; RETURN WITH FLAGS RESTORED
523 <1>
524 <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
525 <1>
526 <1> ;--- CMOS_WRITE -----
527 <1> ; WRITE BYTE TO CMOS_SYSTEM CLOCK CONFIGURATION TABLE :
528 <1> ; :
529 <1> ; INPUT: (AL)= CMOS_TABLE ADDRESS TO BE WRITTEN TO :
530 <1> ; BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT :
531 <1> ; BITS 6-0 = ADDRESS OF TABLE LOCATION TO WRITE :
532 <1> ; (AH)= NEW VALUE TO BE PLACED IN THE ADDRESSED TABLE LOCATION :
533 <1> ; :
534 <1> ; OUTPUT: VALUE IN (AH) PLACED IN LOCATION (AL) WITH NMI LEFT DISABLED :
535 <1> ; IF BIT 7 OF (AL) IS ON, DURING THE CMOS UPDATE BOTH NMI AND :
536 <1> ; NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. :
537 <1> ; THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND :
538 <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:
543 00005B6B 9C <1> pushf ; WRITE (AH) TO LOCATION (AL)
544 00005B6C 6650 <1> push ax ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
545 00005B6E D0C0 <1> rol al, 1 ; SAVE WORK REGISTER VALUES
546 00005B70 F9 <1> stc ; MOVE NMI BIT TO LOW POSITION
547 00005B71 D0D8 <1> rcr al, 1 ; FORCE NMI BIT ON IN CARRY FLAG
548 00005B73 FA <1> cli ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
549 00005B74 E670 <1> out CMOS_PORT, al ; DISABLE INTERRUPTS
550 00005B76 88E0 <1> mov al, ah ; ADDRESS LOCATION AND DISABLE NMI
551 00005B78 E671 <1> out CMOS_DATA, al ; GET THE DATA BYTE TO WRITE
552 00005B7A B01E <1> mov al, CMOS_SHUT_DOWN*2 ; PLACE IN REQUESTED CMOS LOCATION
553 <1> ;mov al, CMOS_REG_D*2 ; GET ADDRESS OF DEFAULT LOCATION
554 00005B7C D0D8 <1> rcr al, 1 ; GET ADDRESS OF DEFAULT LOCATION
555 00005B7E E670 <1> out CMOS_PORT, al ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
556 00005B80 90 <1> nop ; SET DEFAULT TO READ ONLY REGISTER
557 00005B81 E471 <1> in al, CMOS_DATA ; I/O DELAY
558 00005B83 6658 <1> pop ax ; OPEN STANDBY LATCH
559 00005B85 9D <1> popf ; RESTORE WORK REGISTERS
560 00005B86 C3 <1> retn
561 <1>
562 <1> ; /// End Of TIMER FUNCTIONS ///
2160
2161 00005B87 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 00005B90 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 00005B98 FFFF0000009ACF00 dw 0FFFFh, 0, 9A00h, 00CFh ; KCODE
2175 ; 10h kernel data segment, base = 00000000h
2176 00005BA0 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 00005BA8 FFFB000040FACF00 dw 0FBFFh, 0, 0FA40h, 00CFh ; UCODE
2180 ; 23h user data segment, base address = 400000h ; CORE
2181 00005BB0 FFFB000040F2CF00 dw 0FBFFh, 0, 0F240h, 00CFh ; UDATA
2182 ; Task State Segment
2183 00005BB8 6700 dw 0067h ; Limit = 103 ; (104-1, tss size = 104 byte,
2184 ; no IO permission in ring 3)

```

```

2185 gdt_tss0:
2186 dw 0 ; TSS base address, bits 0-15
2187 gdt_tss1:
2188 db 0 ; TSS base address, bits 16-23
2189 ; 49h
2190 db 11101001b ; E9h => P=1/DPL=11/0/1/0/B/1 --> B = Task is busy (1)
2191 db 0 ; G/0/0/AVL/LIMIT=0000 ; (Limit bits 16-19 = 0000) (G=0, 1 byte)
2192 gdt_tss2:
2193 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 gtdt:
2237 dw gdt_end - gdt - 1 ; Limit (size)
2238 dd gdt ; Address of the GDT
2239
2240 ; 20/08/2014
2241 idtd:
2242 dw idt_end - idt - 1 ; Limit (size)
2243 dd idt ; Address of the IDT
2244
2245 ; 20/02/2017
2246 ;; 11/03/2015
2247 %include 'diskdata.s' ; DISK (BIOS) DATA (initialized)
2248
2249 <1> ; *****
2250 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - diskdata.s
2251 <1> ; -----
2252 <1> ; Last Update: 24/01/2016
2253 <1> ; -----
2254 <1> ; Beginning: 24/01/2016
2255 <1> ; -----
2256 <1> ; Assembler: NASM version 2.11 (trdos386.s)
2257 <1> ; -----
2258 <1> ; Turkish Rational DOS
2259 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
2260 <1> ;
2261 <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
2262 <1> ; diskdata.inc (11/03/2015)
2263 <1> ;
2264 <1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
2265 <1> ; *****
2266 <1> ;
2267 <1> ; Retro UNIX 386 v1 Kernel - DISKDATA.INC
2268 <1> ; Last Modification: 11/03/2015
2269 <1> ; (Initialized Disk Parameters Data section for 'DISKIO.INC')
2270 <1> ;
2271 <1> ;
2272 <1> ; -----
2273 <1> ; 80286 INTERRUPT LOCATIONS :
2274 <1> ; REFERENCED BY POST & BIOS :
2275 <1> ; -----
2276 <1> ;
2277 <1> DISK_POINTER: dd MD_TBL6 ; Pointer to Diskette Parameter Table
2278 <1> ;
2279 <1> ; IBM PC-XT Model 286 source code ORGS.ASM (06/10/85) - 14/12/2014
2280 <1> ; -----
2281 <1> ; DISK_BASE :
2282 <1> ; THIS IS THE SET OF PARAMETERS REQUIRED FOR :
2283 <1> ; DISKETTE OPERATION. THEY ARE POINTED AT BY THE :
2284 <1> ; DATA VARIABLE @DISK_POINTER. TO MODIFY THE PARAMETERS, :
2285 <1> ; BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT :
2286 <1> ; -----
2287 <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>
93 <1> ;-----
94 <1> ; ADDITIONAL MEDIA DATA :
95 <1> ;-----
96 <1>
97 <1> ;@LAstrate DB ? ; LAST DISKETTE DATA RATE SELECTED
98 <1> ;@DSK_STATE DB ? ; DRIVE 0 MEDIA STATE
99 <1> ; DB ? ; DRIVE 1 MEDIA STATE
100 <1> ; DB ? ; DRIVE 0 OPERATION START STATE
101 <1> ; DB ? ; DRIVE 1 OPERATION START STATE
102 <1> ;@DSK_TRK DB ? ; DRIVE 0 PRESENT CYLINDER
103 <1> ; DB ? ; DRIVE 1 PRESENT CYLINDER
104 <1>
105 <1> ;DATA ENDS ; END OF BIOS DATA SEGMENT
106 <1>
107 <1> ;-----
108 <1> ; DRIVE TYPE TABLE :
109 <1> ;-----
110 <1> ; 16/02/2015 (unix386.s, 32 bit modifications)
111 <1> DR_TYPE:
112 00005BD0 01 <1> DB 01 ;DRIVE TYPE, MEDIA TABLE
113 <1> ;DW MD_TBL1
114 00005BD1 [EE5B0000] <1> dd MD_TBL1
115 00005BD5 82 <1> DB 02+BIT7ON
116 <1> ;DW MD_TBL2
117 00005BD6 [FB5B0000] <1> dd MD_TBL2
118 00005BDA 02 <1> DR_DEFAULT: DB 02
119 <1> ;DW MD_TBL3
120 00005BDB [085C0000] <1> dd MD_TBL3
121 00005BDF 03 <1> DB 03
122 <1> ;DW MD_TBL4
123 00005BE0 [155C0000] <1> dd MD_TBL4
124 00005BE4 84 <1> DB 04+BIT7ON
125 <1> ;DW MD_TBL5
126 00005BE5 [225C0000] <1> dd MD_TBL5
127 00005BE9 04 <1> DB 04
128 <1> ;DW MD_TBL6
129 00005BEA [2F5C0000] <1> dd MD_TBL6
130 <1> DR_TYPE_E equ $ ; END OF TABLE
131 <1> ;DR_CNT EQU (DR_TYPE_E-DR_TYPE)/3
132 <1> DR_CNT equ (DR_TYPE_E-DR_TYPE)/5
133 <1> ;-----
134 <1> ; MEDIA/DRIVE PARAMETER TABLES :
135 <1> ;-----
136 <1> ;-----
137 <1> ; 360 KB MEDIA IN 360 KB DRIVE :
138 <1> ;-----
139 <1> MD_TBL1:
140 00005BEE DF <1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
141 00005BEF 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE

```

```
142 00005BF0 25 <1> DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
143 00005BF1 02 <1> DB 2 ; 512 BYTES/SECTOR
144 00005BF2 09 <1> DB 09 ; EOT (LAST SECTOR ON TRACK)
145 00005BF3 2A <1> DB 02AH ; GAP LENGTH
146 00005BF4 FF <1> DB 0FFH ; DTL
147 00005BF5 50 <1> DB 050H ; GAP LENGTH FOR FORMAT
148 00005BF6 F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT
149 00005BF7 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
150 00005BF8 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)
151 00005BF9 27 <1> DB 39 ; MAX. TRACK NUMBER
152 00005BFA 80 <1> DB RATE_250 ; DATA TRANSFER RATE
153 <1> ;-----
154 <1> ; 360 KB MEDIA IN 1.2 MB DRIVE :
155 <1> ;-----
156 <1> MD_TBL2:
157 00005BFB DF <1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
158 00005BFC 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
159 00005BFD 25 <1> DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
160 00005BFE 02 <1> DB 2 ; 512 BYTES/SECTOR
161 00005BFF 09 <1> DB 09 ; EOT (LAST SECTOR ON TRACK)
162 00005C00 2A <1> DB 02AH ; GAP LENGTH
163 00005C01 FF <1> DB 0FFH ; DTL
164 00005C02 50 <1> DB 050H ; GAP LENGTH FOR FORMAT
165 00005C03 F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT
166 00005C04 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
167 00005C05 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)
168 00005C06 27 <1> DB 39 ; MAX. TRACK NUMBER
169 00005C07 40 <1> DB RATE_300 ; DATA TRANSFER RATE
170 <1> ;-----
171 <1> ; 1.2 MB MEDIA IN 1.2 MB DRIVE :
172 <1> ;-----
173 <1> MD_TBL3:
174 00005C08 DF <1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
175 00005C09 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
176 00005C0A 25 <1> DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
177 00005C0B 02 <1> DB 2 ; 512 BYTES/SECTOR
178 00005C0C 0F <1> DB 15 ; EOT (LAST SECTOR ON TRACK)
179 00005C0D 1B <1> DB 01BH ; GAP LENGTH
180 00005C0E FF <1> DB 0FFH ; DTL
181 00005C0F 54 <1> DB 054H ; GAP LENGTH FOR FORMAT
182 00005C10 F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT
183 00005C11 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
184 00005C12 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)
185 00005C13 4F <1> DB 79 ; MAX. TRACK NUMBER
186 00005C14 00 <1> DB RATE_500 ; DATA TRANSFER RATE
187 <1> ;-----
188 <1> ; 720 KB MEDIA IN 720 KB DRIVE :
189 <1> ;-----
190 <1> MD_TBL4:
191 00005C15 DF <1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
192 00005C16 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
193 00005C17 25 <1> DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
194 00005C18 02 <1> DB 2 ; 512 BYTES/SECTOR
195 00005C19 09 <1> DB 09 ; EOT (LAST SECTOR ON TRACK)
196 00005C1A 2A <1> DB 02AH ; GAP LENGTH
197 00005C1B FF <1> DB 0FFH ; DTL
198 00005C1C 50 <1> DB 050H ; GAP LENGTH FOR FORMAT
199 00005C1D F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT
200 00005C1E 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
201 00005C1F 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)
202 00005C20 4F <1> DB 79 ; MAX. TRACK NUMBER
203 00005C21 80 <1> DB RATE_250 ; DATA TRANSFER RATE
204 <1> ;-----
205 <1> ; 720 KB MEDIA IN 1.44 MB DRIVE :
206 <1> ;-----
207 <1> MD_TBL5:
208 00005C22 DF <1> DB 11011111B ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
209 00005C23 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
210 00005C24 25 <1> DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
211 00005C25 02 <1> DB 2 ; 512 BYTES/SECTOR
212 00005C26 09 <1> DB 09 ; EOT (LAST SECTOR ON TRACK)
213 00005C27 2A <1> DB 02AH ; GAP LENGTH
214 00005C28 FF <1> DB 0FFH ; DTL
215 00005C29 50 <1> DB 050H ; GAP LENGTH FOR FORMAT
216 00005C2A F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT
217 00005C2B 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
218 00005C2C 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)
219 00005C2D 4F <1> DB 79 ; MAX. TRACK NUMBER
220 00005C2E 80 <1> DB RATE_250 ; DATA TRANSFER RATE
221 <1> ;-----
222 <1> ; 1.44 MB MEDIA IN 1.44 MB DRIVE :
223 <1> ;-----
224 <1> MD_TBL6:
225 00005C2F AF <1> DB 10101111B ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
226 00005C30 02 <1> DB 2 ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
227 00005C31 25 <1> DB MOTOR_WAIT ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
228 00005C32 02 <1> DB 2 ; 512 BYTES/SECTOR
229 00005C33 12 <1> DB 18 ; EOT (LAST SECTOR ON TRACK)
230 00005C34 1B <1> DB 01BH ; GAP LENGTH
231 00005C35 FF <1> DB 0FFH ; DTL
232 00005C36 6C <1> DB 06CH ; GAP LENGTH FOR FORMAT
233 00005C37 F6 <1> DB 0F6H ; FILL BYTE FOR FORMAT
234 00005C38 0F <1> DB 15 ; HEAD SETTLE TIME (MILLISECONDS)
235 00005C39 08 <1> DB 8 ; MOTOR START TIME (1/8 SECONDS)
236 00005C3A 4F <1> DB 79 ; MAX. TRACK NUMBER
237 00005C3B 00 <1> DB RATE_500 ; DATA TRANSFER RATE
238 <1>
239 <1>
240 <1> ; << diskette.inc >>
241 <1> ; ++++++
242 <1> ;
243 <1> ;-----
```

```

244 <1> ; ROM BIOS DATA AREAS :
245 <1> ;-----
246 <1>
247 <1> ;DATA SEGMENT AT 40H ; ADDRESS= 0040:0000
248 <1>
249 <1> ;-----
250 <1> ; FIXED DISK DATA AREAS :
251 <1> ;-----
252 <1>
253 <1> ;DISK_STATUS1: DB 0 ; FIXED DISK STATUS
254 <1> ;HF_NUM: DB 0 ; COUNT OF FIXED DISK DRIVES
255 <1> ;CONTROL_BYTE: DB 0 ; HEAD CONTROL BYTE
256 <1> ;@PORT_OFF DB ? ; RESERVED (PORT OFFSET)
257 <1>
258 <1> ;-----
259 <1> ; ADDITIONAL MEDIA DATA :
260 <1> ;-----
261 <1>
262 <1> ;@LASTRATE DB ? ; LAST DISKETTE DATA RATE SELECTED
263 <1> ;HF_STATUS DB 0 ; STATUS REGISTER
264 <1> ;HF_ERROR DB 0 ; ERROR REGISTER
265 <1> ;HF_INT_FLAG DB 0 ; FIXED DISK INTERRUPT FLAG
266 <1> ;HF_CNTRL DB 0 ; COMBO FIXED DISK/DISKETTE CARD BIT 0=1
267 <1> ;@DSK_STATE DB ? ; DRIVE 0 MEDIA STATE
268 <1> ; DB ? ; DRIVE 1 MEDIA STATE
269 <1> ; DB ? ; DRIVE 0 OPERATION START STATE
270 <1> ; DB ? ; DRIVE 1 OPERATION START STATE
271 <1> ;@DSK_TRK DB ? ; DRIVE 0 PRESENT CYLINDER
272 <1> ; DB ? ; DRIVE 1 PRESENT CYLINDER
273 <1>
274 <1> ;DATA ENDS ; END OF BIOS DATA SEGMENT
275 <1> ;
276 <1> ; ++++++
277 <1>
278 <1> ERR_TBL:
279 00005C3C E0 <1> db NO_ERR
280 00005C3D 024001BB <1> db BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
281 00005C41 04BB100A <1> db RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
282 <1>
283 <1> ; 17/12/2014 (mov ax, [cfd])
284 <1> ; 11/12/2014
285 00005C45 00 <1> cfd: db 0 ; current floppy drive (for GET_PARM)
286 <1> ; 17/12/2014 ; instead of 'DISK_POINTER'
287 00005C46 01 <1> pfd: db 1 ; previous floppy drive (for GET_PARM)
288 <1> ; (initial value of 'pfd
289 <1> ; must be different then 'cfd' value
290 <1> ; to force updating/initializing
291 <1> ; current drive parameters)
292 00005C47 90 <1> align 2
293 <1>
294 00005C48 F001 <1> HF_PORT: dw 1F0h ; Default = 1F0h
295 <1> ; (170h)
296 00005C4A F603 <1> HF_REG_PORT: dw 3F6h ; HF_PORT + 206h
297 <1>
298 <1> ; 05/01/2015
299 00005C4C 00 <1> hf_m_s: db 0 ; (0 = Master, 1 = Slave)
300 <1>
301 <1> ; *****
2248
2249 00005C4D 90 Align 2
2250
2251 ; 04/11/2014 (Retro UNIX 386 v1)
2252 00005C4E 0000 mem_1m_1k: dw 0 ; Number of contiguous KB between
2253 ; 1 and 16 MB, max. 3C00h = 15 MB.
2254 00005C50 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 00005C52 00 boot_drv: db 0 ; boot drive number (physical)
2259 ; 24/11/2014
2260 00005C53 00 drv: db 0
2261 00005C54 00 last_drv: db 0 ; last hdd
2262 00005C55 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 00005C56 00 fd0_type: db 0 ; floppy drive type
2268 00005C57 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 00005C58 00 hd0_type: db 0 ; EDD status for hd0 (bit 7 = present flag)
2274 00005C59 00 hd1_type: db 0 ; EDD status for hd1 (bit 7 = present flag)
2275 00005C5A 00 hd2_type: db 0 ; EDD status for hd2 (bit 7 = present flag)
2276 00005C5B 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 00005C5C 000000000000000000- drv.cylinders: dw 0,0,0,0,0,0,0
2285 00005C65 000000000000000000-
2286 00005C6A 000000000000000000- drv.heads: dw 0,0,0,0,0,0,0
2286 00005C73 000000000000000000-
2287 00005C78 000000000000000000- drv.spt: dw 0,0,0,0,0,0,0
2287 00005C81 000000000000000000-
2288 00005C86 000000000000000000- drv.size: dd 0,0,0,0,0,0,0

```

```
2288 00005C8F 000000000000000000-
2288 00005C98 000000000000000000-
2288 00005CA1 00
2289 00005CA2 0000000000000000    drv.status:    db 0,0,0,0,0,0,0
2290 00005CA9 0000000000000000    drv.error:    db 0,0,0,0,0,0,0
2291
2292
2293
2294
2295
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35 00005CB0 524F50514B
36 00005CB5 4C4D474849
37
38 00005CBA 101112131415
39 00005CC0 161718191E1F
40 00005CC6 202122232425
41 00005CCC 262C2D2E2F30
42 00005CD2 3132
43
44
45
46 00005CD4 52
47 00005CD5 3A4546381D
48 00005CDA 2A36
49
50
51
52 00005CDC 80
53 00005CDD 4020100804
54 00005CE2 0201
55
56
57 00005CE4 1BFF00FFFFFF
58 00005CEA 1EFFFFFFF1F
59 00005CF0 FF7FFF111705
60 00005CF6 12141915090F
61 00005CFC 101B1D0AFF01
62 00005D02 13040607080A
63 00005D08 0B0CFFFFFFF
64 00005D0E 1C1A18031602
65 00005D14 0E0DFFFFFFF
66 00005D1A 96FF20FF
67
68 00005D1E 5E5F60616263
69 00005D24 64656667FFFF
70 00005D2A 778D848E738F
71 00005D30 749075917692
72 00005D36 93FFFFFF898A
73
74
75 00005D3C 1B3132333435363738-
75 00005D45 39302D3D0809
76 00005D4B 71776572747975696F-
76 00005D54 705B5D0DFF61736466-
76 00005D5D 67686A6B6C3B27
77 00005D64 60FF5C7A786376626E-
77 00005D6D 6D2C2E2FFF2AFF20FF
78
79 00005D76 3B3C3D3E3F
80 00005D7B 4041424344
81 00005D80 FFFF
82
83
84 00005D82 474849FF4BFF
85 00005D88 4DFF4F50515253
86 00005D8F FFFF5C8586
87
88
```

```
Align 2

;;; 11/03/2015
%include 'kybdata.s'      ; KEYBOARD (BIOS) DATA
<1> ; *****
<1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - kybdata.s
<1> ; -----
<1> ; Last Update: 17/01/2016
<1> ; -----
<1> ; Beginning: 17/01/2016
<1> ; -----
<1> ; Assembler: NASM version 2.11 (trdos386.s)
<1> ; -----
<1> ; Turkish Rational DOS
<1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
<1> ;
<1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
<1> ; kybdata.inc (11/03/2015)
<1> ;
<1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
<1> ; *****
<1>
<1> ; Retro UNIX 386 v1 Kernel - KYBDATA.INC
<1> ; Last Modification: 11/03/2015
<1> ; (Data Section for 'KEYBOARD.INC')
<1> ;
<1> ; ////////// KEYBOARD DATA //////////
<1>
<1> ; 05/12/2014
<1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)
<1> ; 03/06/86  KEYBOARD BIOS
<1>
<1> ;-----
<1> ;      KEY IDENTIFICATION SCAN TABLES
<1> ;-----
<1>
<1> ;-----      TABLES FOR ALT CASE -----
<1> ;-----      ALT-INPUT-TABLE
<1> K30:  db      82,79,80,81,75
<1>         db      76,77,71,72,73      ; 10 NUMBER ON KEYPAD
<1> ;-----      SUPER-SHIFT-TABLE
<1>         db      16,17,18,19,20,21  ; A-Z TYPEWRITER CHARS
<1>         db      22,23,24,25,30,31
<1>         db      32,33,34,35,36,37
<1>         db      38,44,45,46,47,48
<1>         db      49,50
<1>
<1> ;-----      TABLE OF SHIFT KEYS AND MASK VALUES
<1> ;-----      KEY_TABLE
<1> _K6:   db      INS_KEY      ; INSERT KEY
<1>         db      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
<1>         db      LEFT_KEY,RIGHT_KEY
<1> _K6L   equ     $_K6
<1>
<1> ;-----      MASK_TABLE
<1> _K7:   db      INS_SHIFT    ; INSERT MODE SHIFT
<1>         db      CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
<1>         db      LEFT_SHIFT,RIGHT_SHIFT
<1>
<1> ;-----      TABLES FOR CTRL CASE      ;---- CHARACTERS -----
<1> _K8:   db      27,-1,0,-1,-1,-1  ; Esc, 1, 2, 3, 4, 5
<1>         db      30,-1,-1,-1,-1,31 ; 6, 7, 8, 9, 0, -
<1>         db      -1,127,-1,17,23,5  ; =, Bksp, Tab, Q, W, E
<1>         db      18,20,25,21,9,15   ; R, T, Y, U, I, O
<1>         db      16,27,29,10,-1,1   ; P, [, ], Enter, Ctrl, A
<1>         db      19,4,6,7,8,10      ; S, D, F, G, H, J
<1>         db      11,12,-1,-1,-1,-1  ; K, L, :, ', `, LShift
<1>         db      28,26,24,3,22,2     ; Bkslash, Z, X, C, V, B
<1>         db      14,13,-1,-1,-1,-1  ; N, M, ,, ., /, RShift
<1>         db      150,-1,' ',-1      ; *, ALT, Spc, CL
<1> ;
<1> ;-----      FUNCTIONS -----
<1>         db      94,95,96,97,98,99  ; F1 - F6
<1>         db      100,101,102,103,-1,-1 ; F7 - F10, NL, SL
<1>         db      119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
<1>         db      116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
<1>         db      147,-1,-1,-1,137,138 ; Del, SysReq, Undef, WT, F11, F12
<1>
<1> ;-----      TABLES FOR LOWER CASE -----
<1> K10:   db      27,'1234567890-','=','8,9
<1>
<1>         db      'qwertyuiop[]',13,-1,'asdfghjkl;',39
<1>
<1>         db      96,-1,92,'zxcvbnm,./',-1,'*','-1,' ',-1
<1>
<1> ;-----      LC TABLE SCAN
<1>         db      59,60,61,62,63      ; BASE STATE OF F1 - F10
<1>         db      64,65,66,67,68
<1>         db      -1,-1      ; NL, SL
<1>
<1> ;-----      KEYPAD TABLE
<1> K15:   db      71,72,73,-1,75,-1  ; BASE STATE OF KEYPAD KEYS
<1>         db      77,-1,79,80,81,82,83
<1>         db      -1,-1,92,133,134    ; SysRq, Undef, WT, F11, F12
<1>
<1> ;-----      TABLES FOR UPPER CASE -----
```



```

89 00005D94 1B21402324255E262A- <1> K11: db 27,'!@#$$',94,'&*()_+',8,0
89 00005D9D 28295F2B0800 <1>
90 00005DA3 51574552545955494F- <1> db 'QWERTYUIOP{}',13,-1,'ASDFGHJKL:''
90 00005DAC 507B7D0DFF41534446- <1>
90 00005DB5 47484A4B4C3A22 <1>
91 00005DBC 7EFF7C5A584356424E- <1> db 126,-1,'|ZXCVBNM<>?',-1,'*',-1,' ',-1
91 00005DC5 4D3C3E3FFF2AFF20FF <1>
92 <1> ;----- UC TABLE SCAN
93 00005DCE 5455565758 <1> K12: db 84,85,86,87,88 ; SHIFTED STATE OF F1 - F10
94 00005DD3 595A5B5C5D <1> db 89,90,91,92,93
95 00005DD8 FFFF <1> db -1,-1 ; NL, SL
96 <1>
97 <1> ;----- NUM STATE TABLE
98 00005DDA 3738392D3435362B31- <1> K14: db '789-456+1230.' ; NUMLOCK STATE OF KEYPAD KEYS
98 00005DE3 3233302E <1>
99 <1> ;
100 00005DE7 FFFF7C8788 <1> db -1,-1,124,135,136 ; SysRq, Undef, WT, F11, F12
101 <1>
102 <1> ; 26/08/2014
103 <1> ; Retro UNIX 8086 v1 - UNIX.ASM (03/03/2014)
104 <1> ; Derived from IBM "pc-at"
105 <1> ; rombios source code (06/10/1985)
106 <1> ; 'dseg.inc'
107 <1>
108 <1> ;-----;
109 <1> ; SYSTEM DATA AREA ;
110 <1> ;-----
111 00005DEC 00 <1> BIOS_BREAK db 0 ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
112 <1>
113 <1> ;-----
114 <1> ; KEYBOARD DATA AREAS ;
115 <1> ;-----
116 <1>
117 00005DED 00 <1> KB_FLAG db 0 ; KEYBOARD SHIFT STATE AND STATUS FLAGS
118 00005DEE 00 <1> KB_FLAG_1 db 0 ; SECOND BYTE OF KEYBOARD STATUS
119 00005DEF 00 <1> KB_FLAG_2 db 0 ; KEYBOARD LED FLAGS
120 00005DF0 00 <1> KB_FLAG_3 db 0 ; KEYBOARD MODE STATE AND TYPE FLAGS
121 00005DF1 00 <1> ALT_INPUT db 0 ; STORAGE FOR ALTERNATE KEY PAD ENTRY
122 00005DF2 [025E0000] <1> BUFFER_START dd KB_BUFFER ; OFFSET OF KEYBOARD BUFFER START
123 00005DF6 [225E0000] <1> BUFFER_END dd KB_BUFFER + 32 ; OFFSET OF END OF BUFFER
124 00005DFA [025E0000] <1> BUFFER_HEAD dd KB_BUFFER ; POINTER TO HEAD OF KEYBOARD BUFFER
125 00005DFE [025E0000] <1> BUFFER_TAIL dd KB_BUFFER ; POINTER TO TAIL OF KEYBOARD BUFFER
126 <1> ; ----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
127 00005E02 0000<rept> <1> KB_BUFFER times 16 dw 0 ; ROOM FOR 16 SCAN CODE ENTRIES
128 <1>
129 <1> ; /// End Of KEYBOARD DATA ///
2296 %include 'vidata.s' ; VIDEO (BIOS) DATA
1 <1> ; *****
2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.0 - vidata.s
3 <1> ; -----
4 <1> ; Last Update: 31/07/2016
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 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
14 <1> ; vidata.inc (11/03/2015)
15 <1> ;
16 <1> ; Derived from 'IBM PC-AT' BIOS source code (1985)
17 <1> ; *****
18 <1>
19 <1> ; Retro UNIX 386 v1 Kernel - VIDATA.S
20 <1> ; Last Modification: 11/03/2015
21 <1> ; (Data section for 'VIDEO.INC')
22 <1> ;
23 <1> ; ////////// VIDEO DATA //////////
24 <1>
25 <1> ;-----
26 <1> ; VIDEO DISPLAY DATA AREA ;
27 <1> ;-----
28 00005E22 03 <1> CRT_MODE: db 3 ; CURRENT DISPLAY MODE (TYPE)
29 00005E23 29 <1> CRT_MODE_SET: db 29h ; CURRENT SETTING OF THE 3X8 REGISTER
30 <1> ; (29h default setting for video mode 3)
31 <1> ; Mode Select register Bits
32 <1> ; BIT 0 - 80x25 (1), 40x25 (0)
33 <1> ; BIT 1 - ALPHA (0), 320x200 GRAPHICS (1)
34 <1> ; BIT 2 - COLOR (0), BW (1)
35 <1> ; BIT 3 - Video Sig. ENABLE (1), DISABLE (0)
36 <1> ; BIT 4 - 640x200 B&W Graphics Mode (1)
37 <1> ; BIT 5 - ALPHA mode BLINKING (1)
38 <1> ; BIT 6, 7 - Not Used
39 <1>
40 <1> ; Mode 0 - 2Ch = 101100b ; 40x25 text, 16 gray colors
41 <1> ; Mode 1 - 28h = 101000b ; 40x25 text, 16 fore colors, 8 back colors
42 <1> ; Mode 2 - 2Dh = 101101b ; 80x25 text, 16 gray colors
43 <1> ; Mode 3 - 29h = 101001b ; 80x25 text, 16 fore color, 8 back color
44 <1> ; Mode 4 - 2Ah = 101010b ; 320x200 graphics, 4 colors
45 <1> ; Mode 5 - 2Eh = 101110b ; 320x200 graphics, 4 gray colors
46 <1> ; Mode 6 - 1Eh = 011110b ; 640x200 graphics, 2 colors
47 <1> ; Mode 7 - 29h = 101001b ; 80x25 text, black & white colors
48 <1> ; Mode & 37h = Video signal OFF
49 <1>
50 <1> ; 24/06/2016
51 00005E24 50 <1> CRT_COLS: db 80 ; Number of columns
52 <1>
53 <1> ; 01/07/2016
54 00005E25 00 <1> CRT_PALETTE: db 0 ; Current palette setting
55 <1>

```

```

56      <1> ; 03/07/2016
57      <1> CHAR_HEIGHT: db      16      ; Default character height
58      <1> VGA_VIDEO_CTL:      db      60h      ; ROM BIOS DATA AREA Offset 87h
59      <1> VGA_SWITCHES:      db      0F9h      ; Feature Bit Switches (the basic screen)
60      <1> VGA_MODESET_CTL: db      051h      ; Basic mode set options (VGA video flags)
61      <1>
62      <1>
63      <1>
64      <1>
65      <1>
66      <1>
67      <1>
68      <1>
69      <1>
70      <1>
71      <1>
72      <1> VGA_ROWS:      db      25
73      <1>
74      <1> ; 16/01/2016
75      <1> chr_attrib:      ; Character color/attributes for viode pages (0 to 7)
76      <1>      db      07h, 07h, 07h, 07h, 07h, 07h, 07h, 07h
77      <1> ; 30/01/2016
78      <1> vmode:
79      <1>      db      3,3,3,3,3,3,3,3 ; video modes for pseudo screens
80      <1>
81      <1> CURSOR_MODE: ; cursor start (ch) = 14, cursor end (cl) = 15
82      <1>      db      15, 14 ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
83      <1>
84      <1> ;align 4
85      <1> ;VGA_BASE: ; 26/07/2016
86      <1> ;      dd      0B8000h ; (Mode < 0Dh) or 0A0000h (mode >= 0Dh)
87      <1>
88      <1> align 2
89      <1>
90      <1> vga_modes:
91      <1>
92      <1>
93      <1>
94      <1> ; valid (implemented) video modes (>7, extension to IBM PC CGA modes)
95      <1>      db      03h, 02h, 01h, 00h, 07h, 04h, 05h, 06h
96      <1> vga_g_modes: ; 31/07/2016
97      <1>      db      13h, 0F0h, 12h, 6Ah, 0Dh, 0Eh, 10h, 11h
98      <1> vga_mode_count equ $ - vga_modes
99      <1> vga_g_mode_count equ $ - vga_g_modes
100     <1>
101     <1> vga_mode_tbl_ptr:
102     <1>      ; 25/07/2016
103     <1>      dd      vga_mode_03h
104     <1>      dd      vga_mode_03h ; mode 02h -> mode 03h
105     <1>      dd      vga_mode_01h
106     <1>      dd      vga_mode_01h ; mode 00h -> mode 01h
107     <1>      ;dd      vga_mode_07h
108     <1>      dd      vga_mode_03h ; mode 07h -> mode 03h
109     <1>      dd      vga_mode_04h
110     <1>      dd      vga_mode_04h ; mode 05h -> mode 04h
111     <1>      dd      vga_mode_06h
112     <1>      dd      vga_mode_13h
113     <1>      dd      vga_mode_F0h
114     <1>      dd      vga_mode_12h
115     <1>      dd      vga_mode_6Ah
116     <1>      dd      vga_mode_0Dh
117     <1>      dd      vga_mode_0Eh
118     <1>      dd      vga_mode_10h
119     <1>      dd      vga_mode_11h
120     <1>
121     <1> vga_memmodel:
122     <1>      ; 25/07/2016
123     <1>      ; 07/07/2016
124     <1>      CTEXT equ 0
125     <1>      ;MTEXT equ 1
126     <1>      MTEXT equ 0 ; mode 07h -> mode 03h
127     <1>      CGA equ 2
128     <1>      LINEAR8 equ 5
129     <1>      PLANAR4 equ 4
130     <1>      PLANAR1 equ 3
131     <1>      db      CTEXT, CTEXT, CTEXT, CTEXT, MTEXT, CGA, CGA, CGA
132     <1> vga_g_memmodel: ; 31/07/2016
133     <1>      db      LINEAR8, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR1
134     <1> ;vga_pixbits:
135     <1> ;      ; 25/07/2016
136     <1> ;      ; 08/07/2016
137     <1> ;      db      4, 4, 4, 4, 4, 2, 2, 1, 8, 4, 4, 4, 4, 4, 4, 1
138     <1> vga_dac_s:
139     <1>      db      2, 2, 2, 2, 0, 1, 1, 1, 3, 3, 2, 2, 1, 1, 2, 2
140     <1>
141     <1> vga_params:
142     <1>      ; 25/07/2016
143     <1>      ; 19/07/2016
144     <1>      ; 03/07/2016
145     <1>      ; derived from 'Plex86/Bochs VGABios' source code
146     <1>      ; vgabios-0.7a (2011)
147     <1>      ; by the LGPL VGABios Developers Team (2001-2008)
148     <1>      ; 'vgatables.h'
149     <1>      ; Oracle VirtualBox 5.0.24 VGABios Source Code
150     <1>      ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
151     <1>      ;
152     <1> vga_mode_03h: ; mode 03h, 80*25 text, CGA colors
153     <1>      db      80, 24, 16, 00h, 10h ; tw, th-1, ch, slength (5)
154     <1>      db      00h, 03h, 00h, 02h ; sequ regs (4)
155     <1>      db      67h ; misc reg (1)
156     <1>      db      5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh

```

```
157 00005EC0 004F <1> db 00h, 4Fh
158 <1> vga_p_cm_pos equ $ - vga_mode_03h
159 00005EC2 0D0E00000000 <1> db 0Dh, 0Eh, 00h, 00h, 00h, 00h
160 00005EC8 9C8E8F281F96B9A3 <1> db 9Ch, 8Eh, 8Fh, 28h, 1Fh, 96h, 0B9h, 0A3h
161 00005ED0 FF <1> db 0FFh ; crtc_regs (25)
162 00005ED1 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
163 00005ED9 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
164 00005EE1 0C000F08 <1> db 0Ch, 00h, 0Fh, 08h ; actl regs (20)
165 00005EE5 0000000000100E0FFF <1> db 00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs (9)
166 <1> vga_mode_01h: ; mode 01h, 40*25 text, CGA colors
167 00005EEE 2818100008 <1> db 40, 24, 16, 00h, 08h ; tw, th-1, ch, slength
168 00005EF3 08030002 <1> db 08h, 03h, 00h, 02h ; sequ regs
169 00005EF7 67 <1> db 67h ; misc reg
170 00005EF8 2D2728902BA0BF1F <1> db 2Dh, 27h, 28h, 90h, 2Bh, 0A0h, 0BFh, 1Fh
171 00005F00 004F0D0E00000000 <1> db 00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h
172 00005F08 9C8E8F141F96B9A3 <1> db 9Ch, 8Eh, 8Fh, 14h, 1Fh, 96h, 0B9h, 0A3h
173 00005F10 FF <1> db 0FFh ; crtc_regs
174 00005F11 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
175 00005F19 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
176 00005F21 0C000F08 <1> db 0Ch, 00h, 0Fh, 08h ; actl regs
177 00005F25 0000000000100E0FFF <1> db 00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs
178 <1> ;vga_mode_07h: ; mode 07h, 80*25 text, mono color
179 <1> ; db 80, 24, 16, 00h, 10h ; tw, th-1, ch, slength
180 <1> ; db 00h, 03h, 00h, 02h ; sequ regs
181 <1> ; db 66h ; misc reg
182 <1> ; db 5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh
183 <1> ; db 00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h
184 <1> ; db 9Ch, 8Eh, 8Fh, 28h, 0Fh, 96h, 0B9h, 0A3h
185 <1> ; db 0FFh ; crtc_regs
186 <1> ; db 00h, 08h, 08h, 08h, 08h, 08h, 08h, 08h
187 <1> ; db 10h, 18h, 18h, 18h, 18h, 18h, 18h, 18h
188 <1> ; db 0Eh, 00h, 0Fh, 08h ; actl regs
189 <1> ; db 00h, 00h, 00h, 00h, 00h, 10h, 0Ah, 0Fh, 0FFh ; grdc regs
190 <1> vga_mode_04h: ; 320*200 graphics, 4 colors, CGA
191 00005F2E 2818080008 <1> db 40, 24, 8, 00h, 08h ; tw, th-1, ch, slength
192 00005F33 09030002 <1> db 09h, 03h, 00h, 02h ; sequ regs
193 00005F37 63 <1> db 63h ; misc reg
194 00005F38 2D2728902B80BF1F <1> db 2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh
195 00005F40 00C1000000000000 <1> db 00h, 0C1h, 00h, 00h, 00h, 00h, 00h, 00h
196 00005F48 9C8E8F140096B9A2 <1> db 9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0A2h
197 00005F50 FF <1> db 0FFh ; crtc_regs
198 00005F51 0013151702040607 <1> db 00h, 13h, 15h, 17h, 02h, 04h, 06h, 07h
199 00005F59 1011121314151617 <1> db 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
200 00005F61 01000300 <1> db 01h, 00h, 03h, 00h ; actl regs
201 00005F65 0000000000300F0FFF <1> db 00h, 00h, 00h, 00h, 00h, 30h, 0Fh, 0Fh, 0FFh ; grdc regs
202 <1> vga_mode_06h: ; 640*200 graphics, 2 colors, CGA
203 00005F6E 5018080010 <1> db 80, 24, 8, 00h, 10h ; tw, th-1, ch, slength
204 00005F73 01010006 <1> db 01h, 01h, 00h, 06h ; sequ regs
205 00005F77 63 <1> db 63h ; misc reg
206 00005F78 5F4F50825480BF1F <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
207 00005F80 00C1000000000000 <1> db 00h, 0C1h, 00h, 00h, 00h, 00h, 00h, 00h
208 00005F88 9C8E8F280096B9C2 <1> db 9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0C2h
209 00005F90 FF <1> db 0FFh ; crtc_regs
210 00005F91 0017171717171717 <1> db 00h, 17h, 17h, 17h, 17h, 17h, 17h, 17h
211 00005F99 1717171717171717 <1> db 17h, 17h, 17h, 17h, 17h, 17h, 17h, 17h
212 00005FA1 01000100 <1> db 01h, 00h, 01, 00h ; actl regs
213 00005FA5 0000000000000D0FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 0Dh, 0Fh, 0FFh ; grdc regs
214 <1> vga_mode_13h: ; mode 13h, 300*200, 256 colors, linear
215 00005FAE 2818080000 <1> db 40, 24, 8, 0, 0 ; tw, th-1, ch, slength (5)
216 00005FB3 010F000E <1> db 01h, 0Fh, 00h, 0Eh ; sequ regs (4)
217 00005FB7 63 <1> db 63h ; misc reg (1)
218 00005FB8 5F4F50825480BF1F <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
219 00005FC0 0041000000000000 <1> db 00h, 41h, 00h, 00h, 00h, 00h, 00h, 00h
220 00005FC8 9C8E8F284096B9A3 <1> db 9Ch, 8Eh, 8Fh, 28h, 40h, 96h, 0B9h, 0A3h
221 00005FD0 FF <1> db 0FFh ; crtc_regs (25)
222 00005FD1 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
223 00005FD9 08090A0B0C0D0E0F <1> db 08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
224 00005FE1 41000F00 <1> db 41h, 00h, 0Fh, 00h ; actl regs (20)
225 00005FE5 000000000040050FFF <1> db 00h, 00h, 00h, 00h, 00h, 40h, 05h, 0Fh, 0FFh ; grdc regs (9)
226 <1> vga_mode_set1 equ $ - vga_mode_13h ; = 64
227 <1> vga_mode_F0h: ; mode X ; 320*240, 256 colors, planar
228 00005FEE 2818080000 <1> db 40, 24, 8, 0, 0 ; tw, th-1, ch, slength
229 00005FF3 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
230 00005FF7 E3 <1> db 0E3h ; misc reg
231 00005FF8 5F4F508254800D3E <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Dh, 3Eh
232 00006000 0041000000000000 <1> db 00h, 41h, 00h, 00h, 00h, 00h, 00h, 00h
233 00006008 EAACDF2800E706E3 <1> db 0EAh, 0ACh, 0DFh, 28h, 00h, 0E7h, 06h, 0E3h
234 00006010 FF <1> db 0FFh ; crtc_regs (25)
235 00006011 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
236 00006019 08090A0B0C0D0E0F <1> db 08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
237 00006021 41000F00 <1> db 41h, 00h, 0Fh, 00h ; actl regs
238 00006025 000000000040050FFF <1> db 00h, 00h, 00h, 00h, 00h, 40h, 05h, 0Fh, 0FFh ; grdc regs
239 <1> vga_mode_12h: ; mode 12h, 640*480, 16 colors, planar
240 0000602E 501D100000 <1> db 80, 29, 16, 0, 0 ; tw, th-1, ch, slength
241 00006033 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
242 00006037 E3 <1> db 0E3h ; misc reg
243 00006038 5F4F508254800B3E <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh
244 00006040 0040000000000000 <1> db 00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
245 00006048 EA8CDF2800E704E3 <1> db 0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h
246 00006050 FF <1> db 0FFh ; crtc_regs
247 00006051 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
248 00006059 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
249 00006061 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
250 00006065 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
251 <1> vga_mode_6Ah: ; mode 6Ah, 800*600, 16 colors, planar
252 0000606E 6424100000 <1> db 100, 36, 16, 0, 0 ; tw, th-1, ch, slength
253 00006073 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
254 00006077 E3 <1> db 0E3h ; misc reg
255 00006078 7F6363836B1B72F0 <1> db 7Fh, 63h, 63h, 83h, 6Bh, 1Bh, 72h, 0F0h
256 00006080 0060000000000000 <1> db 00h, 60h, 00h, 00h, 00h, 00h, 00h, 00h
257 00006088 598D5732005773E3 <1> db 59h, 8Dh, 57h, 32h, 00h, 57h, 73h, 0E3h
258 00006090 FF <1> db 0FFh ; crtc_regs
```

```
259 00006091 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
260 00006099 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
261 000060A1 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
262 000060A5 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
263 <1> vga_mode_0Dh: ; mode 0Dh, 320*200, 16 colors, planar
264 000060AE 2818080020 <1> db 40, 24, 8, 0, 20h ; tw, th-1, ch, slength
265 000060B3 090F0006 <1> db 09h, 0Fh, 00h, 06h ; sequ regs
266 000060B7 63 <1> db 63h ; misc reg
267 000060B8 2D2728902B80BF1F <1> db 2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh
268 000060C0 00C0000000000000 <1> db 00h, 0C0h, 00h, 00h, 00h, 00h, 00h, 00h
269 000060C8 9C8E8F140096B9E3 <1> db 9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0E3h
270 000060D0 FF <1> db 0FFh ; crtc regs
271 000060D1 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
272 000060D9 1011121314151617 <1> db 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
273 000060E1 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
274 000060E5 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
275 <1> vga_mode_0Eh: ; mode 0Eh, 640*200, 16 colors, planar
276 000060EE 5018080040 <1> db 80, 24, 8, 0, 40h ; tw, th-1, ch, slength
277 000060F3 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
278 000060F7 63 <1> db 63h ; misc reg
279 000060F8 5F4F50825480BF1F <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
280 00006100 00C0000000000000 <1> db 00h, 0C0h, 00h, 00h, 00h, 00h, 00h, 00h
281 00006108 9C8E8F280096B9E3 <1> db 9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0E3h
282 00006110 FF <1> db 0FFh ; crtc regs
283 00006111 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
284 00006119 1011121314151617 <1> db 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
285 00006121 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
286 00006125 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
287 <1> vga_mode_10h: ; mode 10h, 640*350, 16 colors, planar
288 0000612E 50180E0080 <1> db 80, 24, 14, 0, 80h ; tw, th-1, ch, slength
289 00006133 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
290 00006137 A3 <1> db 0A3h ; misc reg
291 00006138 5F4F50825480BF1F <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
292 00006140 0040000000000000 <1> db 00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
293 00006148 83855D280F63BAE3 <1> db 83h, 85h, 5Dh, 28h, 0Fh, 63h, 0BAh, 0E3h
294 00006150 FF <1> db 0FFh ; crtc regs
295 00006151 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
296 00006159 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
297 00006161 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
298 00006165 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
299 <1> vga_mode_11h: ; mode 11h, 640*480, mono color, planar
300 0000616E 501D100000 <1> db 80, 29, 16, 0, 0 ; tw, th-1, ch, slength
301 00006173 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
302 00006177 E3 <1> db 0E3h ; misc reg
303 00006178 5F4F508254800B3E <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh
304 00006180 0040000000000000 <1> db 00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
305 00006188 EA8CDF2800E704E3 <1> db 0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h
306 00006190 FF <1> db 0FFh ; crtc regs
307 00006191 003F003F003F003F <1> db 00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh
308 00006199 003F003F003F003F <1> db 00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh
309 000061A1 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
310 000061A5 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
311 <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: 28/02/2017
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> ; 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 address 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> ;sdsegmt 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> ;;sdsegmt 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>
138 <1> %macro sys 1-4
139 <1>           ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
140 <1>           ; 03/09/2015
141 <1>           ; 13/04/2015
142 <1>           ; Retro UNIX 386 v1 system call.

```

```

143         <1>      %if %0 >= 2
144             mov ebx, %2
145         <1>      %if %0 >= 3
146             mov ecx, %3
147         <1>      %if %0 = 4
148             mov edx, %4
149         <1>      %endif
150     <1>      %endif
151 <1>      %endif
152 <1>      mov eax, %1
153 <1>      ;int 30h
154 <1>      int 40h ; TRDOS 386 (TRDOS v2.0)
155 <1> %endmacro
156 <1>
157 <1> ; TRDOS 386 system calls, interrupt number
158 <1> ; 25/12/2016
159 <1> SYSCALL_INT_NUM equ '40' ; '40h'
160 <1>
161 <1> ; 13/05/2015 - ERROR CODES
162 <1> ERR_FILE_NOT_OPEN equ 10 ; 'file not open !' error
163 <1> ERR_FILE_ACCESS equ 11 ; 'permission denied !' error
164 <1> ; 14/05/2015
165 <1> ERR_DIR_ACCESS equ 11 ; 'permission denied !' error
166 <1> ERR_FILE_NOT_FOUND equ 12 ; 'file not found !' error
167 <1> ERR_TOO_MANY_FILES equ 13 ; 'too many open files !' error
168 <1> ERR_DIR_EXISTS equ 14 ; 'directory already exists !' error
169 <1> ; 16/05/2015
170 <1> ERR_DRV_NOT_RDY equ 15 ; 'drive not ready !' error
171 <1> ; 18/05/2015
172 <1> ERR_DEV_NOT_RDY equ 15 ; 'device not ready !' error
173 <1> ERR_DEV_ACCESS equ 11 ; 'permission denied !' error
174 <1> ERR_DEV_NOT_OPEN equ 10 ; 'device not open !' error
175 <1> ; 07/06/2015
176 <1> ERR_FILE_EOF equ 16 ; 'end of file !' error
177 <1> ERR_DEV_VOL_SIZE equ 16 ; 'out of volume !' error
178 <1> ; 09/06/2015
179 <1> ERR_DRV_READ equ 17 ; 'disk read error !'
180 <1> ERR_DRV_WRITE equ 18 ; 'disk write error !'
181 <1> ; 16/06/2015
182 <1> ERR_NOT_DIR equ 19 ; 'not a (valid) directory !' error
183 <1> ERR_FILE_SIZE equ 20 ; 'file size error !'
184 <1> ; 22/06/2015
185 <1> ERR_NOT_SUPERUSER equ 11 ; 'permission denied !' error
186 <1> ERR_NOT_OWNER equ 11 ; 'permission denied !' error
187 <1> ERR_NOT_FILE equ 11 ; 'permission denied !' error
188 <1> ; 23/06/2015
189 <1> ERR_FILE_EXISTS equ 14 ; 'file already exists !' error
190 <1> ERR_DRV_NOT_SAME equ 21 ; 'not same drive !' error
191 <1> ERR_DIR_NOT_FOUND equ 12 ; 'directory not found !' error
192 <1> ERR_NOT_EXECUTABLE equ 22 ; 'not executable file !' error
193 <1> ; 27/06/2015
194 <1> ERR_INV_PARAMETER equ 23 ; 'invalid parameter !' error
195 <1> ERR_INV_DEV_NAME equ 24 ; 'invalid device name !' error
196 <1> ; 29/06/2015
197 <1> ERR_TIME_OUT equ 25 ; 'time out !' error
198 <1> ERR_DEV_NOT_RESP equ 25 ; 'device not responding !' error
199 <1> ; 10/10/2016
200 <1> ERR_INV_FILE_NAME equ 26 ; 'invalid file name !' error
201 <1> ERR_INV_FLAGS equ 23 ; 'invalid flags !' error
202 <1> ; For code compatibility with previous version of TRDOS (2011)
203 <1> ; (Temporary error codes for current TRDOS 386 -2016- version)
204 <1> ERR_NO_MORE_FILES equ 12 ; 'no more files !' error
205 <1> ERR_PATH_NOT_FOUND equ 3 ; 'path not found !' error
206 <1> ; 'dir not found !' ; TRDOS 8086
207 <1> ERR_NOT_FOUND: equ 2 ; 'file not found !' ; TRDOS 8086
208 <1> ERR_DISK_SPACE equ 39 ; 'out of volume !' TRDOS 8086
209 <1> ; 'insufficient disk space !' ; 27h
210 <1> ERR_DISK_WRITE equ 30 ; 'disk write protected !' ; 16/10/2016
211 <1> ERR_ACCESS_DENIED equ 5 ; 'access denied !' ; TRDOS 8086
212 <1> ; 28/02/2017
213 <1> ERR_PERM_DENIED equ 11 ; 'permission denied !' error
214 <1> ; 18/05/2016
215 <1> ERR_MISC equ 27 ; miscellaneous/other errors
216 <1> ; 15/10/2016
217 <1> ; TRDOS 8086 -> TRDOS 386 (0Bh -> 28)
218 <1> ERR_INV_FORMAT equ 28 ; 'invalid format !' error
219 <1> ; TRDOS 8086 -> TRDOS 386 (0Dh -> 29)
220 <1> ERR_INV_DATA equ 29 ; 'invalid data !' error
221 <1> ; TRDOS 8086 -> TRDOS 386 (0Eh -> 20)
222 <1> ERR_ZERO_LENGTH equ 20 ; 'zero length !' error
223 <1> ; TRDOS 8086 -> TRDOS 386 (15h -> 17, 1Dh -> 18, 1Eh -> 17)
224 <1> ERR_DRV_NR_READ equ 17 ; 'drive not ready or read error !'
225 <1> ERR_DRV_NR_WRITE equ 18 ; 'drive not ready or write error !'
226 <1> ; 15/10/2016
227 <1> ERR_INV_PATH_NAME equ 19 ; 'bad path name !' error
228 <1> ERR_BAD_CMD_ARG equ 1 ; 'bad command argument !' ; TRDOS 8086
229 <1> ERR_INV_FNUMBER equ 1 ; 'invalid function number !' ; TRDOS 8086
230 <1> ERR_BIG_FILE equ 8 ; 'big file & out of memory !' ; TRDOS 8086
231 <1> ERR_BIG_DATA equ 8 ; 'big data & out of memory !' ; TRDOS 8086
232 <1> ERR_CLUSTER equ 35 ; 'cluster not available !' ; TRDOS 8086
233 <1> ERR_OUT_OF_MEMORY equ 4 ; 'out of memory !'
234 <1> ; 'insufficient memory !'
235 <1> ERR_P_VIOLATION equ 6 ; 'protection violation !'
236 <1> ERR_PAGE_FAULT equ 224 ; 'page fault !' ; 0E0h
237 <1> ERR_SWP_DISK_READ equ 40
238 <1> ERR_SWP_DISK_NOT_PRESENT equ 41
239 <1> ERR_SWP_SECTOR_NOT_PRESENT equ 42
240 <1> ERR_SWP_NO_FREE_SPACE equ 43
241 <1> ERR_SWP_DISK_WRITE equ 44
242 <1> ERR_SWP_NO_PAGE_TO_SWAP equ 45
243 <1>
244 <1> ; 26/08/2015

```

```

245 <1> ; 24/07/2015
246 <1> ; 24/06/2015
247 <1> MAX_ARG_LEN      equ 256 ; max. length of sys exec arguments
248 <1> ; 01/07/2015
249 <1> MAX_MSG_LEN      equ 255 ; max. msg length for 'sysmsg'
250 <1> ;
251 <1> ; 06/10/2016
252 <1> OPENFILES        equ 10  ; max. number of open files (system)
253 <1> ; 07/10/2016
254 <1> ;NUMOFDEVICES    equ 20  ; max. num of available devices (sys)
255 <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
37 <1> Sectors        equ 13h
38 <1> Media          equ 15h
39 <1> FATSecs        equ 16h
40 <1> SecPerTrack    equ 18h
41 <1> Heads          equ 1Ah
42 <1> Hidden1        equ 1Ch
43 <1> Hidden2        equ 1Eh
44 <1> HugeSec1       equ 20h
45 <1> HugeSec2       equ 22h
46 <1> DriveNumber    equ 24h
47 <1> Reserved1      equ 25h
48 <1> bootsignature  equ 26h
49 <1> VolumeID       equ 27h
50 <1> VolumeLabel    equ 2Bh
51 <1> FileSysType    equ 36h
52 <1> Reserved2      equ 3Eh ; Starting cluster of P2000
53 <1>
54 <1> ; FAT32 BPB Structure
55 <1> FAT32_FAT_Size equ 36
56 <1> FAT32_RootFClust equ 44
57 <1> FAT32_FSInfoSec equ 48
58 <1> FAT32_DrvNum equ 64
59 <1> FAT32_BootSig equ 66
60 <1> FAT32_VolID equ 67
61 <1> FAT32_VolLab equ 71
62 <1> FAT32_FilSysType equ 82
63 <1>
64 <1> ; BIOS Disk Parameters
65 <1> DPDiskNumber  equ 0h
66 <1> DPDType       equ 1h
67 <1> DPReturn      equ 2h
68 <1> DPHeads       equ 3h
69 <1> DPCylinders   equ 4h
70 <1> DPSecPerTrack equ 6h
71 <1> DPDisk        equ 7h
72 <1> DPTableOff    equ 8h
73 <1> DPTableSeg    equ 0Ah
74 <1> DPNumOfSecs   equ 0Ch
75 <1>
76 <1> ; BIOS INT 13h Extensions (LBA extensions)
77 <1> ; Just After DP Data (DPDiskNumber+)
78 <1> DAP_PacketSize equ 10h ; If extensions present, this byte will be >=10h
79 <1> DAP_Reserved1 equ 11h ; Reserved Byte
80 <1> DAP_NumOfBlocks equ 12h ; Value of this byte must be 0 to 127
81 <1> DAP_Reserved2 equ 13h ; Reserved Byte
82 <1> DAP_Destination equ 14h ; Address of Transfer Buffer as SEGMENT:OFFSET
83 <1> DAP_LBA_Address equ 18h ; LBA=(C1*H0+H1)*S0+S1-1
84 <1> ; C1= Selected Cylinder Number
85 <1> ; H0= Number Of Heads (Maximum Head Number + 1)
86 <1> ; H1= Selected Head Number
87 <1> ; S0= Maximum Sector Number
88 <1> ; S1= Selected Sector Number
89 <1> ; QUAD WORD
90 <1> ; DAP_Flat_Destination equ 20h ; 64 bit address, if value in 4h is FFFF:FFFFh

```

```

91      <1> ; QUAD WORD (Also, value in 0h must be 18h)
92      <1> ; TR-DOS will not use 64 bit Flat Address
93      <1>
94      <1> ; INT 13h Function 48h "Get Enhanced Disk Drive Parameters"
95      <1> ; Just After DP Data (DPDiskNumber+)
96      <1> GetDParams_48h equ 20h ; Word. Data Length, must be 26 (1Ah) for short data.
97      <1> GDP_48h_InfoFlag equ 22h ; Word
98      <1> ; Bit 1 = 1 -> The geometry returned in bytes 4-15 is valid.
99      <1> GDP_48h_NumOfPCyls equ 24h ; Double Word. Number physical cylinders.
100     <1> GDP_48h_NumOfPHeads equ 28h ; Double Word. Number of physical heads.
101     <1> GDP_48h_NumOfPSpT equ 2Ch ; Double word. Num of physical sectors per track.
102     <1> GDP_48h_LBA_Sectors equ 30h ; 8 bytes. Number of physical/LBA sectors.
103     <1> GDP_48h_BytesPerSec equ 38h ; Word. Number of bytes in a sector.
104     <1>
105     <1> ; TR-DOS Standalone Program Extensions to the DiskParams Block
106     <1> ; Just After DP Data (DPDiskNumber+)
107     <1> TRDP_CurrentSector equ 3Ah ; DX:AX (LBA)
108     <1> TRDP_SectorCount equ 3Eh ; CX (or Counter)
109     <1>
110     <1>
111     <1> ; DOS Logical Disks
112     <1> LD_Name equ 0
113     <1> LD_DiskType equ 1
114     <1> LD_PhyDrvNo equ 2
115     <1> LD_FATType equ 3
116     <1> LD_FSType equ 4
117     <1> LD_LBAYes equ 5
118     <1> LD_BPB equ 6
119     <1> LD_FATBegin equ 96
120     <1> LD_ROOTBegin equ 100
121     <1> LD_DATABegin equ 104
122     <1> LD_StartSector equ 108
123     <1> LD_TotalSectors equ 112
124     <1> LD_FreeSectors equ 116
125     <1> LD_Clusters equ 120
126     <1> LD_PartitionEntry equ 124
127     <1> LD_DParamEntry equ 125
128     <1> LD_MediaChanged equ 126
129     <1> LD_CDirLevel equ 127
130     <1> LD_CurrentDirectory equ 128
131     <1>
132     <1> ; Singlix FS Extensions to DOS Logical Disks
133     <1> ; 03/01/2010 (LD_BPB compatibility for CHS r/w)
134     <1>
135     <1> LD_FS_Name equ 0
136     <1> LD_FS_DiskType equ 1
137     <1> LD_FS_PhyDrvNo equ 2
138     <1> LD_FS_FATType equ 3
139     <1> LD_FS_FSType equ 4
140     <1> LD_FS_LBAYes equ 5
141     <1> LD_FS_BPB equ 6
142     <1> LD_FS_MediaAttrib equ 6
143     <1> LD_FS_VersionMajor equ 7
144     <1> LD_FS_RootDirD equ 8
145     <1> LD_FS_MATLocation equ 12
146     <1> LD_FS_Reserved1 equ 16 ;1 reserved byte
147     <1> LD_FS_BytesPerSec equ 17 ; LD_BPB + 0Bh
148     <1> LD_FS_Reserved2 equ 19 ;2 reserved byte
149     <1> LD_FS_DATLocation equ 20
150     <1> LD_FS_DATSectors equ 24
151     <1> LD_FS_Reserved3 equ 28 ;3 reserved word
152     <1> LD_FS_SecPerTrack equ 30 ; LD_BPB + 18h
153     <1> LD_FS_NumHeads equ 32 ; LD_BPB + 1Ah
154     <1> LD_FS_UnDelDirD equ 34
155     <1> LD_FS_Reserved4 equ 38 ;4 reserved word
156     <1> LD_FS_VolumeSerial equ 40
157     <1> LD_FS_VolumeName equ 44
158     <1> LD_FS_BeginSector equ 108
159     <1> LD_FS_VolumeSize equ 112
160     <1> LD_FS_FreeSectors equ 116
161     <1> LD_FS_FirstFreeSector equ 120
162     <1> LD_FS_PartitionEntry equ 124
163     <1> LD_FS_DParamEntry equ 125
164     <1> LD_FS_MediaChanged equ 126
165     <1> LD_FS_CDirLevel equ 127
166     <1> LD_FS_CDIR_Converted equ 128
167     <1>
168     <1> ; Valid FAT Types
169     <1> FS_FAT12 equ 1
170     <1> FS_FAT16_CHS equ 2
171     <1> FS_FAT32_CHS equ 3
172     <1> FS_FAT16_LBA equ 4
173     <1> FS_FAT32_LBA equ 5
174     <1>
175     <1> ; Cursor Location
176     <1> CCCpointer equ 0450h ; BIOS data, current cursor column
177     <1> ; FAT Clusters EOC sign
178     <1> FAT12EOC equ 0FFFh
179     <1> FAT16EOC equ 0FFFFh
180     <1> ;FAT32EOC equ 0FFFFFFFFh ; It is not direct usable for 8086 code
181     <1> ; BAD Cluster
182     <1> FAT12BADDC equ 0FF7h
183     <1> FAT16BADDC equ 0FFF7h
184     <1> ;FAT32BADDC equ 0FFFFFFFF7h ; 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_DATScout 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 000061AE B036 <1> mov al, 00110110b ; 36h
30 000061B0 E643 <1> out 43h, al
31 000061B2 31C0 <1> xor eax, eax ; sub al, al ; 0
32 000061B4 E640 <1> out 40h, al ; LB
33 000061B6 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 000061B8 BF00010900 <1> mov edi, Logical_DOSDisks
39 000061BD B980060000 <1> mov ecx, 6656/4 ; 26*256 = 6656 bytes
40 000061C2 F3AB <1> rep stosd ; 1664 times 4 bytes
41 <1>
42 000061C4 B83F3A2F00 <1> mov eax, '?:/'
43 000061C9 A3[E7380100] <1> mov [Current_Dir_Drv], eax
44 <1>
45 <1> ; Logical DRV INIT (only for hard disks)
46 000061CE 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 000061D3 BE00010900 <1> mov esi, Logical_DOSDisks
53 000061D8 B001 <1> mov al, 1 ; Initialization sign (invalid_fd_parameter)
54 000061DA 83C67E <1> add esi, LD_MediaChanged ; Media Change Status = 1 (init needed)
55 000061DD 8806 <1> mov [esi], al ; A:
56 000061DF 81C600010000 <1> add esi, 100h
57 000061E5 8806 <1> mov [esi], al ; B:
58 <1>
59 <1> _current_drive_bootdisk:
60 000061E7 8A15[525C0000] <1> mov dl, [boot_drv] ; physical drive number
61 000061ED 80FAFF <1> cmp dl, 0FFh
62 000061F0 740A <1> je short _last_dos_diskno_check
63 <1> _boot_drive_check:
64 000061F2 80FA80 <1> cmp dl, 80h
65 000061F5 7218 <1> jb short _current_drive_a
66 000061F7 80EA7E <1> sub dl, 7Eh ; C = 2 , D = 3
67 000061FA EB13 <1> jmp short _current_drive_a
68 <1>
69 <1> _last_dos_diskno_check:
70 000061FC 8A15[A7FA0000] <1> mov dl, [Last_DOS_DiskNo]
71 00006202 80FA02 <1> cmp dl, 2
72 00006205 7706 <1> ja short _current_drive_c
73 00006207 7406 <1> je short _current_drive_a
74 00006209 30D2 <1> xor dl, dl ; A:
75 0000620B EB02 <1> jmp short _current_drive_a
76 <1>
77 <1> _current_drive_c:
78 0000620D B202 <1> mov dl, 2 ; C:
79 <1>
80 <1> _current_drive_a:
81 0000620F 8815[535C0000] <1> mov [drv], dl
82 00006215 BE[A9FA0000] <1> mov esi, msg_CRLF_temp
83 0000621A E89E000000 <1> call print_msg
84 <1>
85 0000621F 8A15[535C0000] <1> mov dl, [drv]
86 00006225 E893090000 <1> call change_current_drive
87 0000622A 730C <1> jnc short _start_mainprog
88 <1>
89 <1> _drv_not_ready_error:
90 0000622C BE[64FD0000] <1> mov esi, msgl_drv_not_ready
91 00006231 E887000000 <1> call print_msg
92 00006236 EB63 <1> jmp _end_of_mainprog
93 <1>
94 <1> _start_mainprog:
95 <1> ; 07/01/2017
96 <1> ; 07/05/2016
97 <1> ; 02/05/2016
98 <1> ; 24/04/2016
99 <1> ; Retro UNIX 386 v1, 'sys_init' (u0.s)
100 <1> ; 23/06/2015
101 <1>
102 <1> ; 02/05/2016
103 <1> ; 24/04/2016
104 00006238 66B80100 <1> mov ax, 1
105 0000623C A2[FB4E0100] <1> mov [u.uno], al
106 00006241 66A3[944E0100] <1> mov [mpid], ax
107 00006247 66A3[664B0100] <1> mov [p.pid], ax
108 0000624D A2[F64B0100] <1> mov [p.stat], al
109 00006252 C605[F04E0100]04 <1> mov byte [u.quant], time_count ; 07/01/2017
110 <1> ;
111 00006259 A1[20380100] <1> mov eax, [k_page_dir]

```

```

112 0000625E A3[004F0100] <1> mov [u.pgdir], eax ; reset
113 <1> ;
114 00006263 E811E9FFFF <1> call allocate_page
115 00006268 0F82A3000000 <1> jc panic
116 0000626E A3[FC4E0100] <1> mov [u.upage], eax ; user structure page
117 00006273 A3[064C0100] <1> mov [p.upage], eax
118 00006278 E876E9FFFF <1> call clear_page
119 <1> ;
120 <1> ; 24/08/2015
121 0000627D FE0D[A14E0100] <1> dec byte [sysflg] ; FFh = ready for system call
122 <1> ; 0 = executing a system call
123 <1> ; 13/04/2016
124 <1> ; Clear Environment Variables Page/Area
125 00006283 BF00300900 <1> mov edi, Env_Page ; 93000h
126 00006288 B980000000 <1> mov ecx, Env_Page_Size / 4 ; 512/4 (4096/4)

127 0000628D 31C0 <1> xor eax, eax
128 0000628F F3AB <1> rep stosd
129 <1>
130 <1> ; 14/04/2016
131 00006291 E8C3320000 <1> call mainprog_startup_configuration
132 <1>
133 00006296 E8630A0000 <1> call dos_prompt
134 <1>
135 <1> _end_of_mainprog:
136 0000629B BE[A9FA0000] <1> mov esi, msg_CRLF_temp
137 000062A0 E818000000 <1> call print_msg
138 000062A5 BE[AFFA0000] <1> mov esi, mainprog_Version
139 000062AA E80E000000 <1> call print_msg
140 <1> ; 24/01/2016
141 000062AF 28E4 <1> sub ah, ah
142 000062B1 E860A9FFFF <1> call int16h ; call getch
143 000062B6 E940AEFFFF <1> jmp cpu_reset
144 <1>
145 000062BB EBFE <1> infinitiveloop: jmp short infinitiveloop
146 <1>
147 <1> print_msg:
148 <1> ; 13/05/2016
149 <1> ; 04/01/2016
150 <1> ; 01/07/2015
151 <1> ; 13/03/2015 (Retro UNIX 386 v1)
152 <1> ; 07/03/2014 (Retro UNIX 8086 v1)
153 <1> ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
154 <1> ;
155 000062BD 8A3D[4E380100] <1> mov bh, [ACTIVE_PAGE] ; 04/01/2016 (ptty)
156 <1> ;mov bl, 07h ; Black background, light gray forecolor
157 <1>
158 000062C3 AC <1> lodsb
159 <1> pmsg1:
160 000062C4 56 <1> push esi
161 <1> ;mov bh, [ACTIVE_PAGE] ; 04/01/2016 (ptty)
162 000062C5 B307 <1> mov bl, 07h ; Black background, light gray forecolor
163 000062C7 E8E6B9FFFF <1> call _write_tty
164 000062CC 5E <1> pop esi
165 000062CD AC <1> lodsb
166 000062CE 20C0 <1> and al, al
167 000062D0 75F2 <1> jnz short pmsg1
168 000062D2 C3 <1> retn
169 <1>
170 <1> clear_screen:
171 <1> ; 13/05/2016
172 <1> ; 30/01/2016
173 <1> ; 24/01/2016
174 <1> ; 04/01/2016
175 000062D3 0FB61D[4E380100] <1> movzx ebx, byte [ACTIVE_PAGE] ; video page number (0 to 7)
176 000062DA 8AA3[335E0000] <1> mov ah, [ebx+vmode] ; default = 03h (80x25 text)
177 000062E0 80FC04 <1> cmp ah, 4
178 000062E3 7205 <1> jb short cls1
179 000062E5 80FC07 <1> cmp ah, 7
180 000062E8 7526 <1> jne short vga_clear
181 <1> cls1:
182 <1> ;mov bh, bl
183 <1> ;mov bl, 7
184 000062EA 3A25[225E0000] <1> cmp ah, [CRT_MODE] ; current video mode ?
185 <1> ;je short cls2 ; yes (current video mode = 3)
186 <1> ;;call set_mode_3 ; set video mode to 3 (& clear screen)
187 <1> ;;retn
188 <1> ;jmp set_mode_3
189 000062F0 0F85C6B9FFFF <1> jne set_mode_3
190 <1> cls2:
191 000062F6 88DF <1> mov bh, bl ; video page (0 to 7)
192 000062F8 B307 <1> mov bl, 07h ; attribute to be used on blanked line
193 000062FA 28C0 <1> sub al, al ; 0 = entire window
194 000062FC 6631C9 <1> xor cx, cx
195 000062FF 66BA4F18 <1> mov dx, 184Fh
196 00006303 E802B7FFFF <1> call _scroll_up ; 24/01/2016
197 <1> ;
198 <1> ;mov bh, [ACTIVE_PAGE] ; video page number (0 to 7)
199 00006308 6631D2 <1> xor dx, dx
200 0000630B E838BAFFFF <1> call _set_cpos ; 24/01/2016
201 <1> ;retn
202 <1> vga_clear:
203 00006310 C3 <1> retn
204 <1>
205 <1> panic:
206 <1> ; 13/05/2016 (TRDOS 386 = TRDOS v2)
207 <1> ; 13/03/2015 (Retro UNIX 386 v1)
208 <1> ; 07/03/2014 (Retro UNIX 8086 v1)
209 00006311 BE[16070100] <1> mov esi, panic_msg
210 00006316 E8A2FFFFF <1> call print_msg
211 <1> key_to_reboot:
212 <1> ; 24/01/2016

```

```

213 0000631B 28E4      <1>      sub     ah, ah
214 0000631D E8F4A8FFFF <1>      call    int16h ; call  getch
215                      <1>      ; wait for a character from the current tty
216                      <1>      ;
217 00006322 B00A      <1>      mov     al, 0Ah
218 00006324 8A3D[4E380100] <1>      mov     bh, [ptty] ; [ACTIVE_PAGE]
219 0000632A B307      <1>      mov     bl, 07h ; Black background,
220                      <1>      ; light gray forecolor
221 0000632C E881B9FFFF <1>      call    _write_tty
222 00006331 E9C5ADFFFF <1>      jmp     cpu_reset
223                      <1>
224                      <1> ctrlbrk:
225                      <1>      ; 12/11/2015
226                      <1>      ; 13/03/2015 (Retro UNIX 386 v1)
227                      <1>      ; 06/12/2013 (Retro UNIX 8086 v1)
228                      <1>      ;
229                      <1>      ; INT 1Bh (control+break) handler
230                      <1>      ;
231                      <1>      ; Retro Unix 8086 v1 feature only!
232                      <1>      ;
233 00006336 66833D[F24E0100]00 <1>      cmp     word [u.intr], 0
234 0000633E 7645      <1>      jna     short cbrk4
235                      <1> cbrk0:
236                      <1>      ; 12/11/2015
237                      <1>      ; 06/12/2013
238 00006340 66833D[F44E0100]00 <1>      cmp     word [u.quit], 0
239 00006348 743B      <1>      jz      short cbrk4
240                      <1>      ;
241                      <1>      ; 20/09/2013
242 0000634A 6650      <1>      push    ax
243 0000634C A0[4E380100] <1>      mov     al, [ptty]
244                      <1>      ;
245                      <1>      ; 12/11/2015
246                      <1>      ;
247                      <1>      ; ctrl+break (EOT, CTRL+D) from serial port
248                      <1>      ; or ctrl+break from console (pseudo) tty
249                      <1>      ; (!redirection!)
250                      <1>      ;
251 00006351 3C08      <1>      cmp     al, 8 ; serial port tty nums > 7
252 00006353 7211      <1>      jb      short cbrk1 ; console (pseudo) tty
253                      <1>      ;
254                      <1>      ; Serial port interrupt handler sets [ptty]
255                      <1>      ; to the port's tty number (as temporary).
256                      <1>      ;
257                      <1>      ; If active process is using a stdin or
258                      <1>      ; stdout redirection (by the shell),
259                      <1>      ; console tty keyboard must be available
260                      <1>      ; to terminate running process,
261                      <1>      ; in order to prevent a deadlock.
262                      <1>      ;
263 00006355 52         <1>      push    edx
264 00006356 0FB615[FB4E0100] <1>      movzx   edx, byte [u.uno]
265 0000635D 3A82[C54B0100] <1>      cmp     al, [edx+p.ttyc-1] ; console tty (rw)
266 00006363 5A         <1>      pop     edx
267 00006364 7412      <1>      je      short cbrk2
268                      <1> cbrk1:
269 00006366 FEC0      <1>      inc     al ; [u.ttyp] : 1 based tty number
270                      <1>      ; 06/12/2013
271 00006368 3A05[DC4E0100] <1>      cmp     al, [u.ttyp] ; recent open tty (r)
272 0000636E 7408      <1>      je      short cbrk2
273 00006370 3A05[DD4E0100] <1>      cmp     al, [u.ttyp+1] ; recent open tty (w)
274 00006376 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 00006378 6631C0      <1>      xor     ax, ax ; 0
282 0000637B 6648      <1>      dec     ax
283                      <1>      ; 0FFFFh = 'ctrl+brk' keystroke
284 0000637D 66A3[F44E0100] <1>      mov     [u.quit], ax
285                      <1> cbrk3:
286 00006383 6658      <1>      pop     ax
287                      <1> cbrk4:
288 00006385 C3         <1>      retn
2305                      <1> %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 00006386 0FB60D[BC380100] <1>      movzx   ecx, byte [HF_NUM] ; number of fixed disks
25 0000638D 80F901      <1>      cmp     cl, 1

```



```

26 00006390 7301      <1>      jnb     short load_hd_partition_tables
27                    <1>      ; No hard disks
28 00006392 C3        <1>      retn
29                    <1>      load_hd_partition_tables:
30 00006393 8B35[C0380100] <1>      mov     esi, [HDPM_TBL_VEC] ; primary master disk FDPT
31 00006399 BF[E63C0100] <1>      mov     edi, PTable_hd0
32 0000639E B280      <1>      mov     dl, 80h
33                    <1>      load_next_hd_partition_table:
34 000063A0 51        <1>      push    ecx
35 000063A1 57        <1>      push    edi
36 000063A2 56        <1>      push    esi ; FDPT (+ DPTE) address
37 000063A3 8A4614    <1>      mov     al, [esi+20] ; DPTE offset 4
38 000063A6 2440      <1>      and     al, 40h ; LBA bit (bit 6)
39                    <1>      ;shr     al, 6
40 000063A8 A2[E73E0100] <1>      mov     [HD_LBA_yes], al
41 000063AD E81C040000 <1>      call    load_masterboot
42 000063B2 7275      <1>      jc      short pass_pt_this_hard_disk
43                    <1>
44 000063B4 BE[A43C0100] <1>      mov     esi, PartitionTable
45 000063B9 89F3      <1>      mov     ebx, esi
46                    <1>      ;mov     ecx, 16
47 000063BB B110      <1>      mov     cl, 16
48 000063BD F3A5      <1>      rep     movsd
49 000063BF 89DE      <1>      mov     esi, ebx
50 000063C1 C605[555C0000]04 <1>      mov     byte [hdc], 4 ; 4 - partition index
51                    <1>      loc_validate_hdp_partition:
52 000063C8 807E0400    <1>      cmp     byte [esi+ptFileSystemID], 0
53 000063CC 7641      <1>      jna     short loc_validate_next_hdp_partition2
54 000063CE 56        <1>      push    esi ; Masterboot partition table offset
55 000063CF 52        <1>      push    edx ; dl = Physical drive number
56 000063D0 FE05[E83E0100] <1>      inc     byte [PP_Counter]
57 000063D6 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 000063D8 E879010000 <1>      call    validate_hd_fat_partition
63 000063DD 730A      <1>      jnc     short loc_set_valid_hdp_partition_entry
64                    <1>      ;pop     edx
65                    <1>      ;push    edx
66 000063DF 8B1424    <1>      mov     edx, [esp]
67 000063E2 E8C5020000 <1>      call    validate_hd_fs_partition
68 000063E7 7224      <1>      jc      short loc_validate_next_hdp_partition1
69                    <1>      loc_set_valid_hdp_partition_entry:
70 000063E9 8A0D[A7FA0000] <1>      mov     cl, [Last_DOS_DiskNo]
71 000063EF 80C141    <1>      add     cl, 'A'
72                    <1>      ; ESI = Logical dos drive description table address
73 000063F2 880E      <1>      mov     [esi+LD_Name], cl
74 000063F4 8A6602    <1>      mov     ah, [esi+LD_PhyDrvNo]
75 000063F7 88E0      <1>      mov     al, ah ; Physical drive number
76 000063F9 2C80      <1>      sub     al, 80h
77 000063FB C0E002    <1>      shl     al, 2
78 000063FE 0404      <1>      add     al, 4 ; 0 Based
79 00006400 2A05[555C0000] <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 00006406 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 00006409 6689467C    <1>      mov     [esi+LD_PartitionEntry], ax
89                    <1>      loc_validate_next_hdp_partition1:
90 0000640D 5A        <1>      pop     edx ; dl = Physical drive number
91 0000640E 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 0000640F FE0D[555C0000] <1>      dec     byte [hdc] ; 4 - partition index
96 00006415 7412      <1>      jz      short pass_pt_this_hard_disk
97 00006417 83C610    <1>      add     esi, 16 ; 10h
98 0000641A EBAC      <1>      jmp     short loc_validate_hdp_partition
99                    <1>      loc_next_hd_partition_table:
100 0000641C FEC2      <1>      inc     dl
101 0000641E 83C620    <1>      add     esi, 32 ; next FDPT address
102 00006421 83C740    <1>      add     edi, 64 ; next partition table destination
103 00006424 E977FFFFFF <1>      jmp     load_next_hd_partition_table
104                    <1>      pass_pt_this_hard_disk:
105 00006429 5E        <1>      pop     esi ; FDPT (+ DPTE) address
106 0000642A 5F        <1>      pop     edi ; Ptable_hd?
107 0000642B 59        <1>      pop     ecx
108 0000642C E2EE      <1>      loop    loc_next_hd_partition_table
109 0000642E 803D[E83E0100]01 <1>      cmp     byte [PP_Counter], 1
110 00006435 7301      <1>      jnb     short load_extended_dos_partitions
111                    <1>      ; Empty partition table
112 00006437 C3        <1>      retn
113                    <1>      load_extended_dos_partitions:
114 00006438 BE[E63C0100] <1>      mov     esi, PTable_hd0
115 0000643D BF[E63D0100] <1>      mov     edi, PTable_ep0
116 00006442 C605[555C0000]80 <1>      mov     byte [hdc], 80h
117                    <1>      next_hd_extd_partition:
118 00006449 56        <1>      push    esi ; PTable_hd? offset
119 0000644A 57        <1>      push    edi ; PTable_ep?
120                    <1>      ;mov     ecx, 4
121 0000644B B104      <1>      mov     cl, 4
122 0000644D 8A15[555C0000] <1>      mov     dl, byte [hdc]
123                    <1>      hd_check_fs_id_05h:
124 00006453 8A4604    <1>      mov     al, [esi+ptFileSystemID]
125 00006456 3C05      <1>      cmp     al, 05h ; Is it an extended dos partition ?
126 00006458 7404      <1>      je      short loc_set_ep_start_sector
127 0000645A 3C0F      <1>      cmp     al, 0Fh ; Is it an extended win4 (LBA mode) partition ?

```

```

128 0000645C 7546      <1>      jne      short continue_to_check_ep
129                  <1> loc_set_ep_start_sector:
130 0000645E FE05[E93E0100] <1>      inc      byte [EP_Counter]
131 00006464 88D4      <1>      mov      ah, dl ; byte [hdc]
132 00006466 86E0      <1>      xchg     ah, al ; al = Drv Number, ah = Partition Identifier
133 00006468 50        <1>      push     eax
134 00006469 30E4      <1>      xor      ah, ah
135 0000646B 2C80      <1>      sub      al, 80h
136 0000646D 50        <1>      push     eax
137 0000646E C0E002    <1>      shl      al, 2 ; al = al * 4
138 00006471 0FB6D8    <1>      movzx    ebx, al
139 00006474 81C3[EA3E0100] <1>      add      ebx, EP_StartSector
140 0000647A 8B4608    <1>      mov      eax, [esi+ptStartSector]
141                  <1>      ; EAX = Extended partition's start sector
142 0000647D 8903      <1>      mov      [ebx], eax
143 0000647F 58        <1>      pop      eax ; AL = Drv number - 80h, AH = 0
144 00006480 5A        <1>      pop      edx ; DL = Drv number, DH = Partition ID
145 00006481 BB[E63A0100] <1>      mov      ebx, MasterBootBuff
146 00006486 803D[E73E0100]01 <1>      cmp      byte [HD_LBA_yes], 1 ; LBA ready = Yes
147 0000648D 7240      <1>      jnb      short loc_hd_load_ep_05h
148 0000648F 80FE05    <1>      cmp      dh, 05h
149 00006492 743B      <1>      je       short loc_hd_load_ep_05h
150                  <1> loc_hd_load_ep_0Fh:
151                  <1>      ; 04/01/2016
152 00006494 51        <1>      push     ecx
153 00006495 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 00006498 B41B      <1>      mov      ah, 1Bh ; LBA read
159 0000649A B001      <1>      mov      al, 1 ; sector count
160 0000649C E865DDFFFF    <1>      call     int13h
161 000064A1 59        <1>      pop      ecx
162 000064A2 733F      <1>      jnc      short loc_hd_move_ep_table
163                  <1> continue_to_check_ep:
164 000064A4 83C610    <1>      add      esi, 16
165 000064A7 E2AA      <1>      loop     hd_check_fs_id_05h
166                  <1> continue_check_ep_next_disk:
167 000064A9 5F        <1>      pop      edi ; PTable_ep?
168 000064AA 5E        <1>      pop      esi ; PTable_hd?
169 000064AB A0[BC380100] <1>      mov      al, [HF_NUM] ; number of hard disks
170 000064B0 047F      <1>      add      al, 7Fh
171 000064B2 3805[555C0000] <1>      cmp      [hdc], al
172 000064B8 0F8392000000 <1>      jnb      loc_validating_hd_partitions_ok
173 000064BE 83C640    <1>      add      esi, 64
174 000064C1 83C740    <1>      add      edi, 64
175 000064C4 FE05[555C0000] <1>      inc      byte [hdc]
176 000064CA E97AFFFFFF    <1>      jmp      next_hd_extd_partition
177                  <1> loc_hd_load_ep_05h:
178 000064CF 51        <1>      push     ecx
179 000064D0 8A7601    <1>      mov      dh, [esi+ptBeginHead]
180 000064D3 668B4E02    <1>      mov      cx, word [esi+ptBeginSector]
181 000064D7 66B80102    <1>      mov      ax, 0201h ; Read 1 sector
182                  <1>      ;mov     ebx, MasterBootBuff
183 000064DB E826DDFFFF    <1>      call     int13h
184 000064E0 59        <1>      pop      ecx
185 000064E1 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 000064E3 8B3C24    <1>      mov      edi, [esp]
190 000064E6 BE[A43C0100] <1>      mov      esi, PartitionTable ; Extended
191 000064EB 89F3      <1>      mov      ebx, esi
192                  <1>      ;mov     ecx, 16
193 000064ED B110      <1>      mov      cl, 16
194 000064EF F3A5      <1>      rep      movsd
195 000064F1 89DE      <1>      mov      esi, ebx
196                  <1> loc_set_hde_sub_partition_count:
197 000064F3 C605[E83E0100]04 <1>      mov      byte [PP_Counter], 4
198                  <1> loc_validate_hde_partition:
199 000064FA 807E0400    <1>      cmp      byte [esi+ptFileSystemID], 0
200 000064FE 763F      <1>      jna      short loc_validate_next_hde_partition2
201 00006500 56        <1>      push     esi ; Extended partition table offset
202 00006501 8A15[555C0000] <1>      mov      dl, byte [hdc]
203 00006507 0FB6C2    <1>      movzx    eax, dl
204 0000650A 2C80      <1>      sub      al, 80h
205 0000650C 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 0000650F 88C1      <1>      mov      cl, al
211 00006511 80C104    <1>      add      cl, 4
212 00006514 2A0D[E83E0100] <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 0000651A 88D5      <1>      mov      ch, dl
217 0000651C 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 0000651F 51        <1>      push     ecx ; *
222 00006520 BF[EA3E0100] <1>      mov      edi, EP_StartSector
223 00006525 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 00006527 E82A000000    <1>      call     validate_hd_fat_partition
228 0000652C 59        <1>      pop      ecx ; *
229 0000652D 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 0000652F 66894E7C      <1>      mov  [esi+LD_PartitionEntry], cx
236      <1>      ;
237 00006533 8A0D[A7FA0000] <1>      mov  cl, [Last_DOS_DiskNo]
238 00006539 80C141      <1>      add  cl, 'A'
239 0000653C 880E      <1>      mov  [esi+LD_Name], cl
240      <1> loc_validate_next_hde_partition1:
241 0000653E 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 0000653F FE0D[E83E0100] <1>      dec  byte [PP_Counter]
246 00006545 0F845EFFFFFFF <1>      jz   continue_check_ep_next_disk
247 0000654B 83C610      <1>      add  esi, 16 ; 10h
248 0000654E EBAA      <1>      jmp  short loc_validate_hde_partition
249      <1> loc_validating_hd_partitions_ok:
250 00006550 A0[A7FA0000] <1>      mov  al, [Last_DOS_DiskNo]
251      <1> loc_drv_init_retn:
252 00006555 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 00006556 8A6604      <1>      mov  ah, [esi+ptFileSystemID]
275 00006559 80FC06      <1>      cmp  ah, 06h ; FAT16 CHS partition
276      <1>      ; 12/02/2016
277      <1>      ;jb  short loc_not_a_valid_fat_partition2
278 0000655C 7305      <1>      jnb  short vhdnp_FAT16_32
279      <1>      ;
280 0000655E 80FC04      <1>      cmp  ah, 04h ; FAT16 CHS partition (< 32MB)
281 00006561 7519      <1>      jne  short loc_not_a_valid_fat_partition1
282      <1> vhdnp_FAT16_32:
283 00006563 B002      <1>      mov  al, 2
284 00006565 7417      <1>      je   short loc_set_valid_hd_partition_params
285 00006567 80FC0E      <1>      cmp  ah, 0Eh ; FAT16 LBA partition
286 0000656A 7710      <1>      ja   short loc_not_a_valid_fat_partition1
287 0000656C 7410      <1>      je   short loc_set_valid_hd_partition_params
288      <1>
289 0000656E FEC0      <1>      inc  al ; 3
290 00006570 80FC0B      <1>      cmp  ah, 0Bh ; FAT32 CHS partition
291 00006573 7409      <1>      je   short loc_set_valid_hd_partition_params
292 00006575 7206      <1>      jnb  short loc_not_a_valid_fat_partition2
293 00006577 80FC0C      <1>      cmp  ah, 0Ch ; FAT32 LBA partition
294 0000657A 7402      <1>      je   short loc_set_valid_hd_partition_params
295      <1> loc_not_a_valid_fat_partition1:
296 0000657C F9      <1>      stc
297      <1> loc_not_a_valid_fat_partition2:
298 0000657D C3      <1>      retn
299      <1>
300      <1> loc_set_valid_hd_partition_params:
301 0000657E FE05[A7FA0000] <1>      inc  byte [Last_DOS_DiskNo] ; > 1
302      <1>      ;
303 00006584 31DB      <1>      xor  ebx, ebx
304 00006586 8A3D[A7FA0000] <1>      mov  bh, [Last_DOS_DiskNo] ; * 256
305 0000658C 81C300010900 <1>      add  ebx, Logical_DOSDisks
306      <1>      ;
307 00006592 C6430102 <1>      mov  byte [ebx+LD_DiskType], 2
308 00006596 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 00006599 66894303 <1>      mov  word [ebx+LD_FATType], ax
312      <1>      ;
313 0000659D 8B4E08      <1>      mov  ecx, [esi+ptStartSector]
314 000065A0 09FF      <1>      or   edi, edi
315 000065A2 7402      <1>      jz   short pass_hd_FAT_ep_start_sector_adding
316      <1> loc_add_hd_FAT_ep_start_sector:
317 000065A4 030F      <1>      add  ecx, [edi]
318      <1> pass_hd_FAT_ep_start_sector_adding:
319 000065A6 894B6C      <1>      mov  [ebx+LD_StartSector], ecx
320      <1> loc_hd_FAT_logical_drv_init:
321 000065A9 89DD      <1>      mov  ebp, ebx
322      <1>      ;mov  dl, [ebx+LD_PhyDrvNo]
323 000065AB A0[E73E0100] <1>      mov  al, [HD_LBA_yes] ; 07/01/2016
324 000065B0 884305      <1>      mov  [ebx+LD_LBAYes], al
325 000065B3 BB[FA3E0100] <1>      mov  ebx, DOSBootSectorBuff ; buffer address
326 000065B8 08C0      <1>      or   al, al
327 000065BA 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 000065BC B41B      <1>      mov     ah, 1Bh ; LBA read
336 000065BE B001      <1>      mov     al, 1 ; sector count
337 000065C0 E841DCFFFF <1>      call    int13h
338 000065C5 7313      <1>      jnc     short loc_hd_drv_FAT_boot_validation
339      <1>      loc_not_a_valid_fat_partition3:
340 000065C7 C3         <1>      retn
341      <1>      loc_hd_FAT_drv_init_load_bs_chs:
342 000065C8 8A7601      <1>      mov     dh, [esi+ptBeginHead]
343 000065CB 668B4E02    <1>      mov     cx, [esi+ptBeginSector]
344 000065CF 66B80102    <1>      mov     ax, 0201h ; Read 1 sector
345      <1>      ;mov     ebx, DOSBootSectorBuff
346 000065D3 E82EDCFFFF <1>      call    int13h
347 000065D8 72ED      <1>      jc      short loc_not_a_valid_fat_partition3
348      <1>      loc_hd_drv_FAT_boot_validation:
349      <1>      ;mov     esi, DOSBootSectorBuff
350 000065DA 89DE      <1>      mov     esi, ebx
351 000065DC 6681BEFE01000055AA <1>      cmp     word [esi+BS_Validation], 0AA55h
352 000065E5 751A      <1>      jne     short loc_not_a_valid_fat_partition4
353 000065E7 807E15F8    <1>      cmp     byte [esi+BPB_Media], 0F8h
354 000065EB 7514      <1>      jne     short loc_not_a_valid_fat_partition4
355 000065ED 66837E1600    <1>      cmp     word [esi+BPB_FATSz16], 0
356 000065F2 770F      <1>      ja      short loc_hd_FAT16_BPB
357 000065F4 807E4229    <1>      cmp     byte [esi+BS_FAT32_BootSig], 29h
358 000065F8 7507      <1>      jne     short loc_not_a_valid_fat_partition4
359      <1>      loc_hd_FAT32_BPB:
360 000065FA B92D000000    <1>      mov     ecx, 45
361 000065FF EB0D      <1>      jmp     short loc_hd_move_FAT_BPB
362      <1>      ;
363      <1>      loc_not_a_valid_fat_partition4:
364 00006601 F9         <1>      stc
365 00006602 C3         <1>      retn
366      <1>      ;
367      <1>      loc_hd_FAT16_BPB:
368 00006603 807E2629    <1>      cmp     byte [esi+BS_BootSig], 29h
369 00006607 75F8      <1>      jne     short loc_not_a_valid_fat_partition4
370 00006609 B920000000    <1>      mov     ecx, 32
371      <1>      loc_hd_move_FAT_BPB:
372 0000660E 89EF      <1>      mov     edi, ebp
373      <1>      ;mov     esi, ebx ; Boot sector
374 00006610 57         <1>      push    edi
375 00006611 83C706      <1>      add     edi, LD_BPB
376 00006614 F366A5      <1>      rep     movsw
377 00006617 5E         <1>      pop     esi
378 00006618 0FB74614    <1>      movzx   eax, word [esi+LD_BPB+BPB_RsvdSecCnt]
379 0000661C 03466C      <1>      add     eax, [esi+LD_StartSector]
380 0000661F 894660      <1>      mov     [esi+LD_FATBegin], eax
381 00006622 807E0303    <1>      cmp     byte [esi+LD_FATType], 3
382 00006626 7224      <1>      jb      short loc_set_FAT16_RootDirLoc
383      <1>      loc_set_FAT32_RootDirLoc:
384 00006628 8B462A      <1>      mov     eax, [esi+LD_BPB+BPB_FATSz32]
385 0000662B 0FB65E16    <1>      movzx   ebx, byte [esi+LD_BPB+BPB_NumFATs]
386 0000662F F7E3      <1>      mul     ebx
387 00006631 034660      <1>      add     eax, [esi+LD_FATBegin]
388      <1>      loc_set_FAT32_data_begin:
389 00006634 894668      <1>      mov     [esi+LD_DATABegin], eax
390 00006637 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 0000663A 8B4632      <1>      mov     eax, [esi+LD_BPB+BPB_RootClus]
395 0000663D 83E802      <1>      sub     eax, 2
396 00006640 7442      <1>      jz      short short loc_set_32bit_FAT_total_sectors
397      <1>      ;movzx   ebx, byte [esi+LD_BPB+BPB_SecPerClust]
398 00006642 8A5E13      <1>      mov     bl, byte [esi+LD_BPB+BPB_SecPerClust]
399 00006645 F7E3      <1>      mul     ebx
400 00006647 014664      <1>      add     [esi+LD_ROOTBegin], eax
401 0000664A EB38      <1>      jmp     short loc_set_32bit_FAT_total_sectors
402      <1>      ;
403      <1>      loc_set_FAT16_RootDirLoc:
404 0000664C 0FB64616    <1>      movzx   eax, byte [esi+LD_BPB+BPB_NumFATs]
405 00006650 0FB7561C    <1>      movzx   edx, word [esi+LD_BPB+BPB_FATSz16]
406 00006654 F7E2      <1>      mul     edx
407 00006656 034660      <1>      add     eax, [esi+LD_FATBegin]
408 00006659 894664      <1>      mov     [esi+LD_ROOTBegin], eax
409      <1>      loc_set_FAT16_data_begin:
410 0000665C 894668      <1>      mov     [esi+LD_DATABegin], eax
411 0000665F B820000000    <1>      mov     eax, 20h ; Size of a directory entry
412      <1>      ;movzx   edx, word [esi+LD_BPB+BPB_RootEntCnt]
413 00006664 668B5617    <1>      mov     dx, [esi+LD_BPB+BPB_RootEntCnt]
414 00006668 F7E2      <1>      mul     edx
415      <1>      ;mov     ecx, 511
416 0000666A 66B9FF01    <1>      mov     cx, 511
417 0000666E 01C8      <1>      add     eax, ecx
418 00006670 41         <1>      inc     ecx ; 512
419 00006671 F7F1      <1>      div     ecx
420 00006673 014668      <1>      add     [esi+LD_DATABegin], eax
421 00006676 0FB74619    <1>      movzx   eax, word [esi+LD_BPB+BPB_TotalSec16]
422 0000667A 6685C0      <1>      test    ax, ax
423 0000667D 7405      <1>      jz      short loc_set_32bit_FAT_total_sectors
424      <1>      loc_set_16bit_FAT_total_sectors:
425 0000667F 894670      <1>      mov     [esi+LD_TotalSectors], eax
426 00006682 EB06      <1>      jmp     short loc_set_hd_FAT_cluster_count
427      <1>      loc_set_32bit_FAT_total_sectors:
428 00006684 8B4626      <1>      mov     eax, [esi+LD_BPB+BPB_TotalSec32]
429 00006687 894670      <1>      mov     [esi+LD_TotalSectors], eax
430      <1>      loc_set_hd_FAT_cluster_count:
431 0000668A 8B5668      <1>      mov     edx, [esi+LD_DATABegin]
432 0000668D 2B566C      <1>      sub     edx, [esi+LD_StartSector]
433 00006690 29D0      <1>      sub     eax, edx

```



```

434 00006692 31D2      <1>      xor     edx, edx ; 0
435 00006694 0FB64E13  <1>      movzx   ecx, byte [esi+LD_BPB+BPB_SecPerClust]
436 00006698 F7F1      <1>      div     ecx
437 0000669A 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 0000669D E859010000 <1>      call    get_free_FAT_sectors
443 000066A2 7207      <1>      jc     short loc_validate_hd_FAT_partition_retn
444 000066A4 894674      <1>      mov     [esi+LD_FreeSectors], eax
445 000066A7 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 000066AB 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 000066AC 8A6604      <1>      mov     ah, [esi+ptFileSystemID]
472 000066AF 80FCA1      <1>      cmp     ah, 0A1h ; SINGLIX FS1 (trfs1) partition
473 000066B2 7549      <1>      jne     short loc_validate_hd_fs_partition_stc_retn
474                                <1> loc_set_valid_hd_fs_partition_params:
475 000066B4 FE05[A7FA0000] <1>      inc     byte [Last_DOS_DiskNo] ; > 1
476 000066BA 30C0      <1>      xor     al, al ; mov al, 0
477                                <1>      ;mov     [drv], dl
478 000066BC 29DB      <1>      sub     ebx, ebx ; 0
479 000066BE 8A3D[A7FA0000] <1>      mov     bh, [Last_DOS_DiskNo]
480 000066C4 81C300010900 <1>      add     ebx, Logical_DOSDisks
481 000066CA C6430102 <1>      mov     byte [ebx+LD_DiskType], 2
482 000066CE 885302      <1>      mov     [ebx+LD_PhyDrvNo], dl
483                                <1>      ;mov     [ebx+LD_FATType], al ; 0
484                                <1>      ;mov     [ebx+LD_FSType], ah
485 000066D1 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 000066D5 89DD      <1>      mov     ebp, ebx ; 10/01/2016
490                                <1>      ;mov     dl, [ebx+LD_PhyDrvNo]
491 000066D7 A0[E73E0100] <1>      mov     al, [HD_LBA_yes] ; 10/01/2016
492 000066DC 884305      <1>      mov     [ebx+LD_LBAYes], al
493 000066DF 89DE      <1>      mov     esi, ebx
494 000066E1 BB[FA3E0100] <1>      mov     ebx, DOSBootSectorBuff ; buffer address
495 000066E6 08C0      <1>      or      al, al
496 000066E8 7515      <1>      jnz     short loc_hd_fs_drv_init_load_bs_lba
497                                <1> loc_hd_fs_drv_init_load_bs_chs:
498 000066EA 8A7601      <1>      mov     dh, [esi+ptBeginHead]
499 000066ED 668B4E02 <1>      mov     cx, [esi+ptBeginSector]
500 000066F1 66B80102 <1>      mov     ax, 0201h ; Read 1 sector
501                                <1>      ;mov     ebx, DOSBootSectorBuff
502 000066F5 E80CDBFFFF <1>      call    int13h
503 000066FA 7311      <1>      jnc     short loc_hd_drv_fs_boot_validation
504                                <1> loc_validate_hd_fs_partition_err_retn:
505 000066FC C3      <1>      retn
506                                <1> loc_validate_hd_fs_partition_stc_retn:
507 000066FD F9      <1>      stc
508 000066FE 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 000066FF 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 00006702 B41B      <1>      mov     ah, 1Bh ; LBA read
518 00006704 B001      <1>      mov     al, 1 ; sector count
519 00006706 E8FBDAFFFF <1>      call    int13h
520 0000670B 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 0000670D 89DE      <1>      mov     esi, ebx ; Boot sector buffer
524 0000670F 6681BEFE01000055AA <1>      cmp     word [esi+BS_Validation], 0AA55h
525 00006718 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 0000671A 66817E035346 <1>      cmp     word [esi+bs_FS_Identifier], 'SF'
529 00006720 75DB      <1>      jne     short loc_validate_hd_fs_partition_stc_retn
530                                <1>      ; 'Alh' check is not necessary
531                                <1>      ; if 'FS' check is passed as OK/Yes.
532 00006722 807E09A1 <1>      cmp     byte [esi+bs_FS_PartitionID], 0A1h
533 00006726 75D5      <1>      jne     short loc_validate_hd_fs_partition_stc_retn
534                                <1>      ;
535 00006728 89EF      <1>      mov     edi, ebp ; 10/01/2016

```

```

536      <1>      ;
537      <1>      mov     al, byte [esi+bs_FS_LBA_Ready]
538      <1>      mov     [edi+LD_FS_LBAYes], al
539      <1>      ;
540      <1>      ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix
541      <1>      mov     al, [esi+bs_FS_MediaAttrib]
542      <1>      mov     byte [edi+LD_FS_MediaAttrib], al
543      <1>      ;
544      <1>      mov     al, [esi+bs_FS_VersionMaj]
545      <1>      mov     [edi+LD_FS_VersionMajor], al
546      <1>      ;
547      <1>      mov     ax, [esi+bs_FS_BytesPerSec]
548      <1>      mov     [edi+LD_FS_BytesPerSec], ax
549      <1>      mov     al, [esi+bs_FS_SecPerTrack]
550      <1>      cbw
551      <1>      mov     [edi+LD_FS_SecPerTrack], ax
552      <1>      mov     al, [esi+bs_FS_Heads]
553      <1>      ;cbw
554      <1>      mov     [edi+LD_FS_NumHeads], ax
555      <1>      ;
556      <1>      mov     eax, [esi+bs_FS_UnDelDirD]
557      <1>      mov     [edi+LD_FS_UnDelDirD], eax
558      <1>      mov     edx, [esi+bs_FS_MATLocation]
559      <1>      mov     [edi+LD_FS_MATLocation], edx
560      <1>      mov     eax, [esi+bs_FS_RootDirD]
561      <1>      mov     [edi+LD_FS_RootDirD], eax
562      <1>      mov     eax, [esi+bs_FS_BeginSector]
563      <1>      mov     [edi+LD_FS_BeginSector], eax
564      <1>      mov     eax, [edi+bs_FS_VolumeSize]
565      <1>      mov     [edi+LD_FS_VolumeSize], eax
566      <1>      ;
567      <1>      mov     eax, edx ; [edi+LD_FS_MATLocation]
568      <1>      add     eax, [edi+LD_FS_BeginSector]
569      <1>      mov     esi, edi
570      <1>      mread_hd_fs_MAT_sector:
571      <1>      ;mov     ebx, DOSBootSectorBuff
572      <1>      mov     ecx, 1
573      <1>      call     disk_read
574      <1>      jc      short loc_validate_hd_fs_partition_retn
575      <1>      ; EDI will not be changed
576      <1>      mov     esi, ebx
577      <1>      use_hdfs_mat_sector_params:
578      <1>      mov     eax, [esi+FS_MAT_DATLocation]
579      <1>      mov     [edi+LD_FS_DATLocation], eax
580      <1>      mov     eax, [esi+FS_MAT_DATScout]
581      <1>      mov     [edi+LD_FS_DATSectors], eax
582      <1>      mov     eax, [esi+FS_MAT_FreeSectors]
583      <1>      mov     [edi+LD_FS_FreeSectors], eax
584      <1>      mov     eax, [esi+FS_MAT_FirstFreeSector]
585      <1>      mov     [edi+LD_FS_FirstFreeSector], eax
586      <1>      mov     eax, [edi+LD_FS_RootDirD]
587      <1>      add     eax, [edi+LD_FS_BeginSector]
588      <1>      mov     esi, edi
589      <1>      read_hd_fs_RDT_sector:
590      <1>      mov     ebx, DOSBootSectorBuff
591      <1>      ;mov     ecx, 1
592      <1>      mov     cl, 1
593      <1>      call     disk_read
594      <1>      jc      short loc_validate_hd_fs_partition_retn
595      <1>      ; EDI will not be changed
596      <1>      mov     esi, ebx
597      <1>      use_hdfs_RDT_sector_params:
598      <1>      mov     eax, [esi+FS_RDT_VolumeSerialNo]
599      <1>      mov     [edi+LD_FS_VolumeSerial], eax
600      <1>      push     edi
601      <1>      ;mov     ecx, 16
602      <1>      mov     cl, 16
603      <1>      add     esi, FS_RDT_VolumeName
604      <1>      add     edi, LD_FS_VolumeName
605      <1>      rep     movsd ; 64 bytes
606      <1>      pop     esi
607      <1>      ; Volume Name Reset
608      <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      <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      <1>      mov     ah, 0Dh ; Alternate disk reset
622      <1>      call     int13h
623      <1>      jnc     short pass_reset_error
624      <1>      harddisk_error:
625      <1>      retn
626      <1>      pass_reset_error:
627      <1>      mov     ebx, MasterBootBuff
628      <1>      mov     ax, 0201h
629      <1>      mov     cx, 1
630      <1>      xor     dh, dh
631      <1>      call     int13h
632      <1>      jc      short harddisk_error
633      <1>      ;
634      <1>      cmp     word [MBIDCode], 0AA55h
635      <1>      ;
636      <1>      je      short load_masterboot_ok
637      <1>      stc

```

```

637      <1> load_masterboot_ok:
638      <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      000067FB 31C0      <1>         xor     eax, eax
653      <1>         ;mov     [esi+LD_FreeSectors], eax ; Reset
654      <1>
655      000067FD 807E0302  <1>         cmp     byte [esi+LD_FATType], 2
656      00006801 7650      <1>         jna     short loc_gfc_get_fat_free_clusters
657      <1>
658      <1>         ; 29/02/2016
659      00006803 48      <1>         dec     eax ; 0FFFFFFFh
660      00006804 89463A      <1>         mov     [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count (reset)
661      00006807 89463E      <1>         mov     [esi+LD_BPB+BPB_Reserved+4], eax ; First Free Cluster (reset)
662      0000680A 40      <1>         inc     eax ; 0
663      <1>         ;
664      0000680B 668B4636      <1>         mov     ax, [esi+LD_BPB+BPB_FSInfo]
665      0000680F 03466C      <1>         add     eax, [esi+LD_StartSector]
666      <1>
667      00006812 BB[FA3E0100] <1>         mov     ebx, DOSBootSectorBuff
668      00006817 B901000000      <1>         mov     ecx, 1
669      0000681C E8B6890000      <1>         call    disk_read
670      00006821 7301      <1>         jnc     short loc_gfc_check_fsinfo_signs
671      <1> retn_gfc_get_fsinfo_sec:
672      00006823 C3      <1>         retn
673      <1>
674      <1> loc_gfc_check_fsinfo_signs:
675      00006824 BB[FA3E0100] <1>         mov     ebx, DOSBootSectorBuff ; 13/02/2016
676      00006829 813B52526141      <1>         cmp     dword [ebx], 41615252h
677      0000682F 7520      <1>         jne     short retn_gfc_get_fsinfo_stc
678      <1>         ;add     ebx, 484
679      <1>         ;cmp     dword [ebx], 61417272h
680      00006831 81BBE4010000727241- <1>         cmp     dword [ebx+484], 61417272h
680      0000683A 61      <1>
681      0000683B 7514      <1>         jne     short retn_gfc_get_fsinfo_stc
682      <1>         ;add     ebx, 4
683      <1>         ;mov     eax, [ebx]
684      0000683D 8B83E8010000      <1>         mov     eax, [ebx+488]
685      <1>         ; 29/02/2016
686      00006843 89463A      <1>         mov     [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count
687      00006846 8B93EC010000      <1>         mov     edx, [ebx+492]
688      0000684C 89463E      <1>         mov     [esi+LD_BPB+BPB_Reserved+4], eax ; First Free Cluster
689      <1>         ;
690      0000684F EB12      <1>         jmp     short retn_from_get_free_fat32_clusters
691      <1>
692      <1> retn_gfc_get_fsinfo_stc:
693      00006851 F9      <1>         stc
694      00006852 C3      <1>         retn
695      <1>
696      <1> loc_gfc_get_fat_free_clusters:
697      <1>         ;mov     eax, 2
698      00006853 B002      <1>         mov     al, 2
699      <1>         ;mov     [FAT_CurrentCluster], eax
700      <1> loc_gfc_loop_get_next_cluster:
701      00006855 E823500000      <1>         call    get_next_cluster
702      0000685A 730E      <1>         jnc     short loc_gfc_free_fat_clusters_cont
703      0000685C 21C0      <1>         and     eax, eax
704      0000685E 7411      <1>         jz      short loc_gfc_pass_inc_free_cluster_count
705      <1>
706      <1> retn_from_get_free_fat_clusters:
707      00006860 8B4674      <1>         mov     eax, [esi+LD_FreeSectors] ; Free clusters !
708      <1> retn_from_get_free_fat32_clusters:
709      00006863 0FB65E13      <1>         movzx    ebx, byte [esi+LD_BPB+BPB_SecPerClust]
710      00006867 F7E3      <1>         mul     ebx
711      <1>         ;mov     [esi+LD_FreeSectors], eax ; Free sectors
712      <1> retn_get_free_sectors_calc:
713      00006869 C3      <1>         retn
714      <1>
715      <1> loc_gfc_free_fat_clusters_cont:
716      0000686A 09C0      <1>         or      eax, eax
717      0000686C 7503      <1>         jnz     short loc_gfc_pass_inc_free_cluster_count
718      0000686E FF4674      <1>         inc     dword [esi+LD_FreeSectors] ; Free clusters !
719      <1>
720      <1> loc_gfc_pass_inc_free_cluster_count:
721      <1>         ;mov     eax, [FAT_CurrentCluster]
722      00006871 89C8      <1>         mov     eax, ecx ; [FAT_CurrentCluster]
723      00006873 3B4678      <1>         cmp     eax, [esi+LD_Clusters]
724      00006876 77E8      <1>         ja      short retn_from_get_free_fat_clusters
725      00006878 40      <1>         inc     eax
726      <1>         ;mov     [FAT_CurrentCluster], eax
727      00006879 EBDA      <1>         jmp     short loc_gfc_loop_get_next_cluster
728      <1>
729      <1> floppy_drv_init:
730      <1>         ; 06/07/2016
731      <1>         ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
732      <1>         ; 24/07/2011
733      <1>         ; 04/07/2009
734      <1>         ; INPUT ->
735      <1>         ;   DL = Drive number (0,1)
736      <1>         ; OUTPUT ->
737      <1>         ;   BL = drive name

```

```

738      <1>      ;      BH = drive number
739      <1>      ;      ESI = Logical DOS drv description table
740      <1>      ;      EAX = Volume serial number
741      <1>
742 0000687B BE[565C0000] <1>      mov     esi, fd0_type ; 10/01/2016
743 00006880 BF00010900 <1>      mov     edi, Logical_DOSDisks
744 00006885 08D2 <1>      or      dl, dl
745 00006887 7407 <1>      jz      short loc_drv_init_fd0_fd1
746 00006889 81C700010000 <1>      add     edi, 100h
747 0000688F 46 <1>      inc     esi ; fd1_type ; 10/01/2016
748 <1> loc_drv_init_fd0_fd1:
749 00006890 C6477E00 <1>      mov     byte [edi+LD_MediaChanged], 0
750 00006894 803E01 <1>      cmp     byte [esi], 1 ; type (>0 if it is existing)
751 <1>      ; 4 = 1.44 MB, 80 track, 3 1/2"
752 00006897 7221 <1>      jnb     short read_fd_boot_sector_retn
753 00006899 885702 <1>      mov     [edi+LD_PhyDrvNo], dl
754 <1> read_fd_boot_sector:
755 0000689C 30F6 <1>      xor     dh, dh
756 0000689E B904000000 <1>      mov     ecx, 4 ; Retry Count
757 <1> read_fd_boot_sector_again:
758 000068A3 51 <1>      push    ecx
759 <1>      ;mov     cx, 1
760 000068A4 B101 <1>      mov     cl, 1
761 000068A6 66B80102 <1>      mov     ax, 0201h ; Read 1 sector
762 000068AA BB[FA3E0100] <1>      mov     ebx, DOSBootSectorBuff
763 000068AF E852D9FFFF <1>      call    int13h
764 000068B4 59 <1>      pop     ecx
765 000068B5 7304 <1>      jnc     short use_fd_boot_sector_params
766 000068B7 E2EA <1>      loop    read_fd_boot_sector_again
767 <1>
768 <1> read_fd_boot_sector_stc_retn:
769 000068B9 F9 <1>      stc
770 <1> read_fd_boot_sector_retn:
771 000068BA C3 <1>      retn
772 <1>
773 <1> use_fd_boot_sector_params:
774 <1>      ;mov     esi, DOSBootSectorBuff
775 000068BB 89DE <1>      mov     esi, ebx
776 000068BD 6681BEFE01000055AA <1>      cmp     word [esi+BS_Validation], 0AA55h
777 000068C6 75F1 <1>      jne     short read_fd_boot_sector_stc_retn
778 000068C8 66817E035346 <1>      cmp     word [esi+bs_FS_Identifier], 'SF'
779 000068CE 0F85A2000000 <1>      jne     use_fd_fatfs_boot_sector_params
780 <1>      ;
781 000068D4 8A462D <1>      mov     al, [esi+bs_FS_LBA_Ready]
782 000068D7 884705 <1>      mov     [edi+LD_FS_LBAYes], al
783 <1>      ;
784 <1>      ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix
785 000068DA 8A4608 <1>      mov     al, [esi+bs_FS_MediaAttrib]
786 000068DD 884706 <1>      mov     [edi+LD_FS_MediaAttrib], al
787 <1>      ;
788 000068E0 8A460A <1>      mov     al, [esi+bs_FS_VersionMaj]
789 000068E3 884707 <1>      mov     byte [edi+LD_FS_VersionMajor], al
790 000068E6 668B4606 <1>      mov     ax, [esi+bs_FS_BytesPerSec]
791 000068EA 66894711 <1>      mov     [edi+LD_FS_BytesPerSec], ax
792 000068EE 8A462E <1>      mov     al, [esi+bs_FS_SecPerTrack]
793 000068F1 6698 <1>      cbw
794 000068F3 6689471E <1>      mov     [edi+LD_FS_SecPerTrack], ax
795 000068F7 8A462F <1>      mov     al, [esi+bs_FS_Heads]
796 <1>      ;cbw
797 000068FA 66894720 <1>      mov     [edi+LD_FS_NumHeads], ax
798 <1>      ;
799 000068FE 8B4628 <1>      mov     eax, [esi+bs_FS_UnDelDirD]
800 00006901 894722 <1>      mov     [edi+LD_FS_UnDelDirD], eax
801 00006904 8B4618 <1>      mov     eax, [esi+bs_FS_MATLocation]
802 00006907 89470C <1>      mov     [edi+LD_FS_MATLocation], eax
803 0000690A 8B461C <1>      mov     eax, [esi+bs_FS_RootDirD]
804 0000690D 894708 <1>      mov     [edi+LD_FS_RootDirD], eax
805 00006910 8B460C <1>      mov     eax, [esi+bs_FS_BeginSector]
806 00006913 89476C <1>      mov     [edi+LD_FS_BeginSector], eax
807 00006916 8B4610 <1>      mov     eax, [esi+bs_FS_VolumeSize]
808 00006919 894770 <1>      mov     [edi+LD_FS_VolumeSize], eax
809 <1>      ;
810 0000691C 89FE <1>      mov     esi, edi
811 0000691E 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 00006921 B101 <1>      mov     cl, 1
817 00006923 E8B5880000 <1>      call    chs_read
818 00006928 89DE <1>      mov     esi, ebx
819 0000692A 7301 <1>      jnc     short use_fdfs_mat_sector_params
820 <1>      ;jmp     short read_fd_boot_sector_retn
821 0000692C C3 <1>      retn
822 <1> use_fdfs_mat_sector_params:
823 0000692D 8B460C <1>      mov     eax, [esi+FS_MAT_DATLocation]
824 00006930 894714 <1>      mov     [edi+LD_FS_DATLocation], eax
825 00006933 8B4610 <1>      mov     eax, [esi+FS_MAT_DATScout]
826 00006936 894718 <1>      mov     [edi+LD_FS_DATSectors], eax
827 00006939 8B4714 <1>      mov     eax, [edi+FS_MAT_FreeSectors]
828 0000693C 894774 <1>      mov     [edi+LD_FS_FreeSectors], eax
829 0000693F 8B4618 <1>      mov     eax, [esi+FS_MAT_FirstFreeSector]
830 00006942 894778 <1>      mov     [edi+LD_FS_FirstFreeSector], eax
831 <1>      ;
832 00006945 89FE <1>      mov     esi, edi
833 00006947 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 0000694A B101 <1>      mov     cl, 1
838 0000694C E88C880000 <1>      call    chs_read
839 00006951 89DE <1>      mov     esi, ebx

```



```

840 00006953 7220      <1>      jc      short read_fd_RDT_sector_retn
841                  <1> use_fdfs_RDT_sector_params:
842 00006955 8B461C      <1>      mov     eax, [esi+FS_RDT_VolumeSerialNo]
843 00006958 894728      <1>      mov     [edi+LD_FS_VolumeSerial], eax
844 0000695B 57          <1>      push    edi
845                  <1>      ;mov     ecx, 16
846 0000695C B110      <1>      mov     cl, 16
847 0000695E 83C640      <1>      add     esi, FS_RDT_VolumeName
848 00006961 83C72C      <1>      add     edi, LD_FS_VolumeName
849 00006964 F3A5      <1>      rep     movsd ; 64 bytes
850 00006966 5E          <1>      pop     esi
851 00006967 C6460300    <1>      mov     byte [esi+LD_FATType], 0
852 0000696B C64604A1    <1>      mov     byte [esi+LD_FSType], 0A1h
853 0000696F E9A5000000    <1>      jmp     loc_cont_use_fd_boot_sector_params
854                  <1>
855                  <1> read_fd_RDT_sector_stc_retn:
856 00006974 F9          <1>      stc
857                  <1> read_fd_RDT_sector_retn:
858 00006975 C3          <1>      retn
859                  <1>
860                  <1> use_fd_fatfs_boot_sector_params:
861 00006976 807E2629    <1>      cmp     byte [esi+BS_BootSig], 29h
862 0000697A 75F8      <1>      jne     short read_fd_RDT_sector_stc_retn
863 0000697C 807E15F0    <1>      cmp     byte [esi+BPB_Media], 0F0h
864 00006980 72F3      <1>      jnb     short read_fd_RDT_sector_retn
865 00006982 57          <1>      push    edi
866 00006983 83C706      <1>      add     edi, LD_BPB
867                  <1>      ;mov     ecx, 16
868 00006986 B110      <1>      mov     cl, 16
869 00006988 F3A5      <1>      rep     movsd ; 64 bytes
870 0000698A 5E          <1>      pop     esi
871 0000698B 31C0      <1>      xor     eax, eax
872 0000698D 89466C      <1>      mov     [esi+LD_StartSector], eax ; 0
873 00006990 668B461C    <1>      mov     ax, [esi+LD_BPB+BPB_FATSz16]
874 00006994 8A4E16      <1>      mov     cl, [esi+LD_BPB+BPB_NumFATs]
875 00006997 F7E1      <1>      mul     ecx
876                  <1>      ; edx = 0 !
877 00006999 668B5614    <1>      mov     dx, [esi+LD_BPB+BPB_RsvdSecCnt]
878 0000699D 66895660    <1>      mov     [esi+LD_FATBegin], dx
879                  <1>      ;add     eax, edx
880                  <1>      add     ax, dx
881 000069A4 894664      <1>      mov     [esi+LD_ROOTBegin], eax
882 000069A7 894668      <1>      mov     [esi+LD_DATABegin], eax
883 000069AA 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
888                  <1>      ;shr     edx, 9 ; edx = ((edx*32)+511) / 512
889                  <1>      ;shr     dx, 9
890 000069AE 6683C20F    <1>      add     dx, 15 ; 06/07/2016 ((512/32)-1)
891 000069B2 66C1EA04    <1>      shr     dx, 4 ; / 16 (==16 entries per sector)
892 000069B6 015668      <1>      add     [esi+LD_DATABegin], edx ; + rd sectors
893                  <1>      ;movzx   eax, word [esi+LD_BPB+BPB_TotalSec16]
894 000069B9 668B4619    <1>      mov     ax, [esi+LD_BPB+BPB_TotalSec16]
895 000069BD 894670      <1>      mov     [esi+LD_TotalSectors], eax
896 000069C0 2B4668      <1>      sub     eax, [esi+LD_DATABegin]
897                  <1>      ;movzx   ecx, byte [esi+LD_BPB+BPB_SecPerClust]
898 000069C3 8A4E13      <1>      mov     cl, [esi+LD_BPB+BPB_SecPerClust]
899 000069C6 80F901      <1>      cmp     cl, 1
900 000069C9 7605      <1>      jna     short save_fd_fatfs_cluster_count
901                  <1>      ;sub     edx, edx
902 000069CB 6629D2      <1>      sub     dx, dx ; 0
903                  <1>      ;sub     dl, dl ; 06/07/2016
904 000069CE F7F1      <1>      div     ecx
905                  <1> save_fd_fatfs_cluster_count:
906 000069D0 894678      <1>      mov     [esi+LD_Clusters], eax
907                  <1>
908                  <1>      ; Maximum Valid Cluster Number = EAX +1
909                  <1>      ; with 2 reserved clusters= EAX +2
910                  <1>
911                  <1> reset_FAT_buffer_decriptors:
912 000069D3 29C0      <1>      sub     eax, eax ; 0
913 000069D5 A2[FE400100] <1>      mov     [FAT_BuffValidData], al ; 0
914 000069DA A2[FF400100] <1>      mov     [FAT_BuffDrvName], al ; 0
915 000069DF A3[02410100] <1>      mov     [FAT_BuffSector], eax ; 0
916                  <1>
917                  <1> read_fd_FAT_sectors:
918 000069E4 BB001C0900    <1>      mov     ebx, FAT_Buffer
919 000069E9 668B4614    <1>      mov     ax, [esi+LD_BPB+BPB_RsvdSecCnt]
920                  <1>      ;mov     ecx, 3
921 000069ED B103      <1>      mov     cl, 3 ; 3 sectors
922 000069EF E8E9870000    <1>      call    chs_read
923 000069F4 7240      <1>      jc     short read_fd_FAT_sectors_retn
924                  <1> use_fd_FAT_sectors:
925 000069F6 8A4602      <1>      mov     al, [esi+LD_PhyDrvNo]
926 000069F9 0441      <1>      add     al, 'A'
927 000069FB A2[FF400100] <1>      mov     [FAT_BuffDrvName], al
928 00006A00 C605[FE400100]01 <1>      mov     byte [FAT_BuffValidData], 1
929 00006A07 E82B000000    <1>      call    fd_init_calculate_free_clusters
930 00006A0C 7228      <1>      jc     short read_fd_FAT_sectors_retn
931                  <1>
932                  <1> loc_use_fd_boot_sector_params_FAT:
933 00006A0E C6460301    <1>      mov     byte [esi+LD_FATType], 1 ; FAT 12
934 00006A12 C6460401    <1>      mov     byte [esi+LD_FSType], 1
935 00006A16 8B462D      <1>      mov     eax, [esi+LD_BPB+VolumeID]
936                  <1> loc_cont_use_fd_boot_sector_params:
937 00006A19 8A7E02      <1>      mov     bh, [esi+LD_PhyDrvNo]
938 00006A1C 887E7D      <1>      mov     [esi+LD_DParamEntry], bh
939 00006A1F 88FB      <1>      mov     bl, bh
940 00006A21 80C341      <1>      add     bl, 'A'
941 00006A24 881E      <1>      mov     byte [esi+LD_Name], bl

```

```

942 00006A26 C6460101      <1>      mov     byte [esi+LD_DiskType], 1
943 00006A2A C6460500      <1>      mov     byte [esi+LD_LBAYes], 0
944 00006A2E C6467C00      <1>      mov     byte [esi+LD_PartitionEntry], 0
945 00006A32 C6467E06      <1>      mov     byte [esi+LD_MediaChanged], 6 ; Volume Name Reset
946                                <1>
947                                <1> read_fd_FAT_sectors_retn:
948 00006A36 C3              <1>      retn
949                                <1>
950                                <1> fd_init_calculate_free_clusters:
951                                <1>      ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
952                                <1>      ; 04/07/2009
953                                <1>      ; INPUT ->
954                                <1>      ;     ESI = Logical DOS drive description table address
955                                <1>      ; OUTPUT ->
956                                <1>      ;     [ESI+LD_FreeSectors] will be set
957                                <1>
958 00006A37 29C0            <1>      sub     eax, eax
959 00006A39 894674          <1>      mov     [esi+LD_FreeSectors], eax ; 0
960 00006A3C B002            <1>      mov     al, 2 ; eax = 2
961                                <1>
962                                <1> fd_init_loop_get_next_cluster:
963 00006A3E E830000000      <1>      call    fd_init_get_next_cluster
964 00006A43 722D            <1>      jc      short fd_init_calculate_free_clusters_retn
965                                <1>
966                                <1> fd_init_free_fat_clusters:
967                                <1>      ;cmp     eax, 0
968                                <1>      ;ja      short fd_init_pass_inc_free_cluster_count
969                                <1>      ;and     eax, eax
970                                <1>      ;jnz     short fd_init_pass_inc_free_cluster_count
971 00006A45 6621C0          <1>      and     ax, ax
972 00006A48 7504            <1>      jnz     short fd_init_pass_inc_free_cluster_count
973                                <1>      ;inc     dword [esi+LD_FreeSectors]
974 00006A4A 66FF4674        <1>      inc     word [esi+LD_FreeSectors]
975                                <1>
976                                <1> fd_init_pass_inc_free_cluster_count:
977                                <1>      ;mov     eax, [FAT_CurrentCluster]
978 00006A4E 66A1[FA400100]  <1>      mov     ax, [FAT_CurrentCluster]
979                                <1>      ;cmp     eax, [esi+LD_Clusters]
980 00006A54 663B4678        <1>      cmp     ax, [esi+LD_Clusters]
981 00006A58 7704            <1>      ja      short short retn_from_fd_init_calculate_free_clusters
982                                <1>      ;inc     eax
983 00006A5A 6640            <1>      inc     ax
984 00006A5C EBEO            <1>      jmp     short fd_init_loop_get_next_cluster
985                                <1>
986                                <1> retn_from_fd_init_calculate_free_clusters:
987 00006A5E 8A4613          <1>      mov     al, [esi+LD_BPB+BPB_SecPerClust]
988 00006A61 3C01            <1>      cmp     al, 1
989 00006A63 760D            <1>      jna     short fd_init_calculate_free_clusters_retn
990                                <1>      ;movzx   eax, al
991 00006A65 6698            <1>      cbw
992                                <1>      ;mov     ecx, [esi+LD_FreeSectors]
993 00006A67 668B4E74        <1>      mov     cx, [esi+LD_FreeSectors] ; Count of free clusters
994                                <1>      ;mul     ecx
995 00006A6B 66F7E1          <1>      mul     cx
996                                <1>      ;mov     [esi+LD_FreeSectors], eax
997 00006A6E 66894674        <1>      mov     [esi+LD_FreeSectors], ax
998                                <1> fd_init_calculate_free_clusters_retn:
999 00006A72 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 00006A73 A3[FA400100]    <1>      mov     [FAT_CurrentCluster], eax
1014                                <1> fd_init_get_next_cluster_readnext:
1015 00006A78 29D2            <1>      sub     edx, edx ; 0
1016 00006A7A BB00040000      <1>      mov     ebx, 1024 ; 400h
1017 00006A7F F7F3            <1>      div     ebx
1018                                <1>      ; EAX = Count of 3 FAT sectors
1019                                <1>      ; EDX = Buffer entry index
1020 00006A81 89C1            <1>      mov     ecx, eax
1021                                <1>      ;mov     eax, 3
1022 00006A83 B003            <1>      mov     al, 3
1023 00006A85 F7E2            <1>      mul     edx ; Multiply by 3
1024 00006A87 66D1E8          <1>      shr     ax, 1 ; Divide by 2
1025 00006A8A 89C3            <1>      mov     ebx, eax ; Buffer byte offset
1026 00006A8C 81C3001C0900    <1>      add     ebx, FAT_Buffer
1027 00006A92 89C8            <1>      mov     eax, ecx
1028                                <1>      ;mov     edx, 3
1029 00006A94 66BA0300        <1>      mov     dx, 3
1030 00006A98 F7E2            <1>      mul     edx
1031                                <1>      ; EAX = FAT Beginning Sector
1032                                <1>      ; EDX = 0
1033 00006A9A 8A0E            <1>      mov     cl, [esi+LD_Name]
1034                                <1>      ;cmp     byte [FAT_BuffValidData], 0
1035                                <1>      ;jna     short fd_init_load_FAT_sectors0
1036 00006A9C 3A0D[FF400100]  <1>      cmp     cl, [FAT_BuffDrvName]
1037 00006AA2 751E            <1>      jne     short fd_init_load_FAT_sectors0
1038 00006AA4 3B05[02410100]  <1>      cmp     eax, [FAT_BuffSector]
1039 00006AAA 751C            <1>      jne     short fd_init_load_FAT_sectors1
1040                                <1>      ;mov     eax, [FAT_CurrentCluster]
1041 00006AAC A0[FA400100]    <1>      mov     al, [FAT_CurrentCluster]
1042                                <1>      ;shr     eax, 1
1043 00006AB1 D0E8            <1>      shr     al, 1

```

```

1044 00006AB3 668B03      <1>      mov     ax, [ebx]
1045 00006AB6 7306      <1>      jnc     short fd_init_gnc_even
1046 00006AB8 66C1E804    <1>      shr     ax, 4
1047      <1> fd_init_gnc_clc_retn:
1048 00006ABC F8      <1>      clc
1049 00006ABD C3      <1>      retn
1050      <1>
1051      <1> fd_init_gnc_even:
1052 00006ABE 80E40F    <1>      and     ah, 0Fh
1053 00006AC1 C3      <1>      retn
1054      <1>
1055      <1> fd_init_load_FAT_sectors0:
1056 00006AC2 880D[FF400100] <1>      mov     [FAT_BuffDrvName], cl
1057      <1> fd_init_load_FAT_sectors1:
1058 00006AC8 C605[FE400100]00 <1>      mov     byte [FAT_BuffValidData], 0
1059 00006ACF A3[02410100]    <1>      mov     [FAT_BuffSector], eax
1060 00006AD4 034660    <1>      add     eax, [esi+LD_FATBegin]
1061 00006AD7 BB001C0900 <1>      mov     ebx, FAT_Buffer
1062      <1> ;movzx ecx, word [esi+LD_BPB+BPB_FATSz16]
1063 00006ADC 668B4E1C    <1>      mov     cx, [esi+LD_BPB+BPB_FATSz16]
1064 00006AE0 662B0D[02410100] <1>      sub     cx, [FAT_BuffSector]
1065      <1> ;cmp ecx, 3
1066 00006AE7 6683F903    <1>      cmp     cx, 3
1067 00006AEB 7605      <1>      jna     short fdinit_pass_fix_sector_count_3
1068      <1> ;mov ecx, 3
1069 00006AED B903000000 <1>      mov     ecx, 3
1070      <1> fdinit_pass_fix_sector_count_3:
1071 00006AF2 E8E6860000    <1>      call    chs_read
1072 00006AF7 730D      <1>      jnc     short fd_init_FAT_sectors_no_load_error
1073 00006AF9 C605[FE400100]00 <1>      mov     byte [FAT_BuffValidData], 0
1074      <1> ; Drv not ready or read Error !
1075 00006B00 B80F000000 <1>      mov     eax, ERR_DRV_NOT_RDY ; 15
1076      <1> ;xor edx, edx
1077 00006B05 C3      <1>      retn
1078      <1>
1079      <1> fd_init_FAT_sectors_no_load_error:
1080 00006B06 C605[FE400100]01 <1>      mov     byte [FAT_BuffValidData], 1
1081 00006B0D A1[FA400100]    <1>      mov     eax, [FAT_CurrentCluster]
1082 00006B12 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      <1> mov     esi, ebx
1100      <1> and     esi, 0FF00h ; esi = bh
1101      <1> add     esi, Logical_DOSDisks
1102      <1> mov     al, [esi+LD_Name]
1103      <1> mov     ah, [esi+LD_FATType]
1104      <1> cmp     ah, 1
1105      <1> jb     short loc_gfvn_dir_load_err
1106      <1> cmp     al, 'A'
1107      <1> jb     short loc_gfvn_dir_load_err
1108      <1> cmp     ah, 2
1109      <1> ja     short get_FAT32_root_cluster
1110      <1>
1111      <1> call    load_FAT_root_directory
1112      <1> jnc     short loc_get_volume_name
1113      <1>
1114      <1> loc_gfvn_dir_load_err:
1115      <1> retn
1116      <1>
1117      <1> get_FAT32_root_cluster:
1118      <1> mov     eax, [esi+LD_BPB+BPB_RootClus]
1119      <1> call    load_FAT_sub_directory
1120      <1> jc     short loc_get_volume_name_retn
1121      <1>
1122      <1> loc_get_volume_name:
1123      <1> mov     esi, Directory_Buffer
1124      <1> xor     cx, cx ; 0
1125      <1> check_root_volume_name:
1126      <1> mov     al, [esi]
1127      <1> or      al, al
1128      <1> jz     short loc_get_volume_name_retn
1129      <1> cmp     byte [esi+0Bh], 08h
1130      <1> je     short loc_get_volume_name_retn
1131      <1> cmp     cx, [DirBuff_LastEntry]
1132      <1> jnb     short pass_check_root_volume_name
1133      <1> inc     cx
1134      <1> add     esi, 32
1135      <1> jmp     short check_root_volume_name
1136      <1>
1137      <1> loc_get_volume_name_retn:
1138      <1> retn
1139      <1>
1140      <1> pass_check_root_volume_name:
1141      <1> cmp     byte [DirBuff_FATType], 3
1142      <1> jb     short loc_get_volume_name_retn_xor
1143      <1>
1144      <1> mov     ebx, FAT_Buffer
1145      <1> mov     esi, Logical_DOSDisks

```

```

1146 00006B82 31C0      <1>      xor     eax, eax
1147 00006B84 8A25[0E410100] <1>      mov     ah, [DirBuff_DRV]
1148 00006B8A 80EC41      <1>      sub     ah, 'A'
1149 00006B8D 01C6      <1>      add     esi, eax
1150 00006B8F A1[15410100] <1>      mov     eax, [DirBuff_Cluster]
1151 00006B94 E8E44C0000 <1>      call    get_next_cluster
1152 00006B99 7305      <1>      jnc     short loc_gfvn_load_FAT32_dir_cluster
1153                                     <1>
1154 00006B9B 83F801      <1>      cmp     eax, 1
1155 00006B9E F5          <1>      cmc
1156 00006B9F C3          <1>      retn
1157                                     <1>
1158                                     <1> loc_gfvn_load_FAT32_dir_cluster:
1159 00006BA0 E8BE4E0000 <1>      call    load_FAT_sub_directory
1160 00006BA5 73A3      <1>      jnc     short loc_get_volume_name
1161 00006BA7 C3          <1>      retn
1162                                     <1>
1163                                     <1> loc_get_volume_name_retn_xor:
1164 00006BA8 31C0      <1>      xor     eax, eax
1165 00006BAA 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 00006BAB B416      <1>      mov     ah, 16h
1177 00006BAD E854D6FFFF <1>      call    int13h
1178 00006BB2 80FC06      <1>      cmp     ah, 06h
1179 00006BB5 7405      <1>      je      short loc_gmc_status_retn
1180 00006BB7 08E4      <1>      or      ah, ah
1181 00006BB9 7401      <1>      jz      short loc_gmc_status_retn
1182                                     <1> loc_gmc_status_stc_retn:
1183 00006BBB F9          <1>      stc
1184                                     <1> loc_gmc_status_retn:
1185 00006BBC C3          <1>      retn
2306                                     <1> %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> ; -----
8                                     <1> ; Assembler: NASM version 2.11 (trdos386.s)
9                                     <1> ; -----
10                                    <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
11                                    <1> ; MAINPROG.ASM (09/11/2011)
12                                    <1> ; *****
13                                    <1> ; MAINPROG.ASM [ TRDOS KERNEL - COMMAND EXECUTER SECTION - MAIN PROGRAM ]
14                                    <1> ; (c) 2004-2011 Erdogan TAN [ 17/01/2004 ] Last Update: 09/11/2011
15                                    <1> ; CMD_INTR.ASM [ TRDOS Command Interpreter Procedure ] Last Update: 09/11/2011
16                                    <1> ; DIR.ASM [ DIRECTORY FUNCTIONS ] Last Update: 09/10/2011
17                                    <1> ; FILE.ASM [ FILE FUNCTIONS ] Last Update: 09/10/2011
18                                    <1>
19                                    <1> change_current_drive:
20                                    <1>      ; 16/10/2016
21                                    <1>      ; 02/02/2016
22                                    <1>      ; 15/01/2016 (TRDOS 386 = TRDOS v2.0)
23                                    <1>      ; 18/08/2011
24                                    <1>      ; 09/09/2009
25                                    <1>      ; INPUT:
26                                    <1>      ;     DL = Logical DOS Drive Number
27                                    <1>      ; OUTPUT:
28                                    <1>      ;     cf=1 -> Not successful
29                                    <1>      ;     EAX = Error code
30                                    <1>      ;     cf=0 ->
31                                    <1>      ;     EAX = 0 (successful)
32                                    <1>
33 00006BBD 31DB      <1>      xor     ebx, ebx
34 00006BBF 88D7      <1>      mov     bh, dl
35                                     <1>
36                                     <1>      ;cmp     dl, 1
37                                     <1>      ;jna     short loc_ccdrv_initial_media_change_check
38                                     <1>      ;cmp     bh, [Last_Dos_DiskNo]
39                                     <1>      ;ja      short loc_ccdrv_drive_not_ready_err
40                                     <1>
41                                     <1> loc_ccdrv_initial_media_change_check:
42 00006BC1 BE00010900 <1>      mov     esi, Logical_DOSDisks
43 00006BC6 01DE      <1>      add     esi, ebx
44                                     <1> loc_ccdrv_dos_drive_name_check:
45 00006BC8 80FA02 <1>      cmp     dl, 2
46 00006BCB 720F      <1>      jnb     short loc_ccdrv_dos_drive_name_check_ok
47                                     <1>
48 00006BCD 8A06      <1>      mov     al, [esi+LD_Name]
49 00006BCF 2C41      <1>      sub     al, 'A'
50 00006BD1 38D0      <1>      cmp     al, dl
51 00006BD3 7407      <1>      je      short loc_ccdrv_dos_drive_name_check_ok
52                                     <1>
53                                     <1> loc_ccdrv_drive_not_ready_err:
54                                     <1>      ; 16/10/2016 (15h -> 15)
55 00006BD5 B80F000000 <1>      mov     eax, 15 ; Drive not ready
56                                     <1> loc_change_current_drive_stc_retn:
57 00006BDA F9          <1>      stc
58 00006BDB C3          <1>      retn
59                                     <1>
60                                     <1> loc_ccdrv_dos_drive_name_check_ok:
61 00006BDC 8A667E <1>      mov     ah, [esi+LD_MediaChanged]

```



```

62 00006BDF 80FC06      <1>      cmp     ah, 6   ; VOLUME NAME CHECK/MOVE SIGN
63 00006BE2 7455      <1>      je      short loc_ccdrv_get_FAT_volume_name_0
64                                <1>
65 00006BE4 80FA01      <1>      cmp     dl, 1
66 00006BE7 777D      <1>      ja      short loc_gmcs_init_drv_hd
67                                <1>
68                                <1> loc_gmcs_init_drv_fd:
69 00006BE9 08E4      <1>      or      ah, ah
70                                <1>      ; AH = 1 is initialization sign (invalid_fd_parameter)
71 00006BEB 7517      <1>      jnz     short loc_ccdrv_call_fd_init
72                                <1>
73 00006BED E8B9FFFFFF      <1>      call    get_media_change_status
74 00006BF2 72E1      <1>      jc      short loc_ccdrv_drive_not_ready_err
75                                <1>
76 00006BF4 20E4      <1>      and     ah, ah
77 00006BF6 7476      <1>      jz      short loc_change_current_drv3
78                                <1>
79 00006BF8 80F406      <1>      xor     ah, 6
80 00006BFB 75D8      <1>      jnz     short loc_ccdrv_drive_not_ready_err
81                                <1>
82                                <1> loc_ccdrv_call_fd_init_check_vol_id:
83 00006BFD E8490A0000      <1>      call    get_volume_serial_number
84 00006C02 730D      <1>      jnc     short loc_ccdrv_check_vol_serial
85                                <1>
86                                <1> loc_ccdrv_call_fd_init:
87 00006C04 E872FCFFFF      <1>      call    floppy_drv_init
88 00006C09 731A      <1>      jnc     short loc_reset_drv_fd_current_dir
89                                <1>
90                                <1> loc_ccdrv_fdinit_fail_retn:
91                                <1>      ; 16/10/2016
92 00006C0B B80F000000      <1>      mov     eax, 15 ; Drive not ready
93 00006C10 C3      <1>      retn
94                                <1>
95                                <1> loc_ccdrv_check_vol_serial:
96 00006C11 A3[DC380100]      <1>      mov     [Current_VolSerial], eax
97                                <1>      ;mov     dl, bh
98 00006C16 E860FCFFFF      <1>      call    floppy_drv_init
99 00006C1B 72EE      <1>      jc      short loc_ccdrv_fdinit_fail_retn
100                               <1>
101 00006C1D 3B05[DC380100]      <1>      cmp     eax, [Current_VolSerial]
102 00006C23 7445      <1>      je      short loc_change_current_drv2
103                               <1>
104                               <1> loc_reset_drv_fd_current_dir:
105 00006C25 31C0      <1>      xor     eax, eax
106 00006C27 88467F      <1>      mov     [esi+LD_CDirLevel], al
107 00006C2A 89F7      <1>      mov     edi, esi
108 00006C2C 81C780000000      <1>      add     edi, LD_CurrentDirectory
109 00006C32 B920000000      <1>      mov     ecx, 32
110 00006C37 F3AB      <1>      rep     stosd
111                               <1>
112                               <1> loc_ccdrv_get_FAT_volume_name_0:
113 00006C39 8A4603      <1>      mov     al, [esi+LD_FATType]
114 00006C3C 08C0      <1>      or      al, al
115 00006C3E 742A      <1>      jz      short loc_change_current_drv2
116                               <1>
117 00006C40 56      <1>      push    esi
118 00006C41 3C02      <1>      cmp     al, 2
119 00006C43 7705      <1>      ja      short loc_ccdrv_get_FAT32_vol_name
120                               <1>
121                               <1> loc_ccdrv_get_FAT2_16_vol_name:
122 00006C45 83C631      <1>      add     esi, LD_BPB + VolumeLabel
123 00006C48 EB03      <1>      jmp     short loc_ccdrv_get_FAT_volume_name_1
124                               <1>
125                               <1> loc_ccdrv_get_FAT32_vol_name:
126 00006C4A 83C64D      <1>      add     esi, LD_BPB + FAT32_VolLab
127                               <1> loc_ccdrv_get_FAT_volume_name_1:
128 00006C4D 53      <1>      push    ebx
129 00006C4E 56      <1>      push    esi
130 00006C4F E8C3FEFFFF      <1>      call    get_FAT_volume_name
131 00006C54 5F      <1>      pop     edi
132 00006C55 5B      <1>      pop     ebx
133                               <1>      ; BL = 0
134 00006C56 720B      <1>      jc      short loc_change_current_drv1
135 00006C58 20C0      <1>      and     al, al
136 00006C5A 7407      <1>      jz      short loc_change_current_drv1
137                               <1>
138                               <1> loc_ccdrv_move_FAT_volume_name:
139 00006C5C B90B000000      <1>      mov     ecx, 11
140 00006C61 F3A4      <1>      rep     movsb
141                               <1>
142                               <1> loc_change_current_drv1:
143 00006C63 5E      <1>      pop     esi
144 00006C64 EB04      <1>      jmp     short loc_change_current_drv2
145                               <1>
146                               <1> loc_gmcs_init_drv_hd:
147 00006C66 08E4      <1>      or      ah, ah
148 00006C68 7404      <1>      jz      short loc_change_current_drv3
149                               <1>      ; BL = 0, BH = Logical DOS drive number
150                               <1> loc_change_current_drv2:
151 00006C6A C6467E00      <1>      mov     byte [esi+LD_MediaChanged], 0
152                               <1> loc_change_current_drv3:
153 00006C6E 883D[E6380100]      <1>      mov     [Current_Drv], bh
154                               <1>
155                               <1> ;call restore_current_directory
156                               <1> ;retn
157                               <1>
158                               <1> restore_current_directory:
159                               <1>      ; 11/02/2016
160                               <1>      ; 15/01/2016 (TRDOS 386 = TRDOS v2.0)
161                               <1>      ; 25/01/2010
162                               <1>      ; 12/10/2009
163                               <1>      ;

```

```

164      <1>      ; INPUT:
165      <1>      ;     ESI = Logical DOS Drive Description Table
166      <1>      ;
167      <1>      ; OUTPUT:
168      <1>      ;     ESI = Logical DOS Drive Description Table
169      <1>      ;     EDI = offset Current_Dir_Drv
170      <1>
171 00006C74 8A4603      <1>      mov     al, [esi+LD_FATType]
172 00006C77 A2[E5380100]      <1>      mov     [Current_FATType], al
173      <1>
174 00006C7C 8A26      <1>      mov     ah, [esi+LD_Name]
175 00006C7E 8825[E7380100] <1>      mov     [Current_Dir_Drv], ah
176      <1>
177 00006C84 20C0      <1>      and     al, al
178 00006C86 741D      <1>      jz      short loc_restore_FS_current_directory
179      <1>
180      <1> loc_restore_FAT_current_directory:
181 00006C88 8A667F      <1>      mov     ah, [esi+LD_CDirLevel]
182 00006C8B 8825[E4380100] <1>      mov     [Current_Dir_Level], ah
183 00006C91 08E4      <1>      or      ah, ah
184 00006C93 7416      <1>      jz      short loc_ccdrv_reset_cdir_FAT_12_16_32_fcluster
185      <1>
186 00006C95 0FB6D4      <1>      movzx   edx, ah
187 00006C98 C0E204      <1>      shl     dl, 4 ; * 16
188 00006C9B 01F2      <1>      add     edx, esi
189 00006C9D 8B828C000000 <1>      mov     eax, [edx+LD_CurrentDirectory+12]
190 00006CA3 EB2C      <1>      jmp     short loc_ccdrv_reset_cdir_FAT_fcluster
191      <1>
192      <1> loc_restore_FS_current_directory:
193 00006CA5 E8F44D0000 <1>      call    load_current_FS_directory
194 00006CAA C3      <1>      retn
195      <1>
196      <1> loc_ccdrv_reset_cdir_FAT_12_16_32_fcluster:
197 00006CAB 3C03      <1>      cmp     al, 3
198 00006CAD 7205      <1>      jb      short loc_ccdrv_reset_cdir_FAT_12_16_fcluster
199      <1> loc_ccdrv_reset_cdir_FAT32_fcluster:
200 00006CAF 8B4632      <1>      mov     eax, [esi+LD_BPB+FAT32_RootFClust]
201 00006CB2 EB04      <1>      jmp     short loc_ccdrv_check_rootdir_sign
202      <1> loc_ccdrv_reset_cdir_FAT_12_16_fcluster:
203 00006CB4 30C0      <1>      xor     al, al ; xor eax, eax
204 00006CB6 31D2      <1>      xor     edx, edx
205      <1> loc_ccdrv_check_rootdir_sign:
206 00006CB8 80BE8000000000 <1>      cmp     byte [esi+LD_CurrentDirectory], 0
207 00006CBF 7510      <1>      jne     short loc_ccdrv_reset_cdir_FAT_fcluster
208      <1> loc_ccdrv_set_rootdir_FAT_fcluster:
209 00006CC1 89868C000000 <1>      mov     [esi+LD_CurrentDirectory+12], eax
210 00006CC7 C78680000000524F4F- <1>      mov     dword [esi+LD_CurrentDirectory], 'ROOT'
210 00006CD0 54      <1>
211      <1>
212      <1> loc_ccdrv_reset_cdir_FAT_fcluster:
213 00006CD1 A3[E0380100] <1>      mov     [Current_Dir_FCluster], eax
214      <1>
215 00006CD6 BF[47410100] <1>      mov     edi, PATH_Array
216 00006CDB 89F2      <1>      mov     edx, esi
217 00006CDD 81C680000000 <1>      add     esi, LD_CurrentDirectory
218 00006CE3 B920000000 <1>      mov     ecx, 32
219 00006CE8 F3A5      <1>      rep     movsd
220      <1>
221 00006CEA E8832D0000 <1>      call    change_prompt_dir_string
222      <1>
223 00006CEF 89D6      <1>      mov     esi, edx
224      <1>
225 00006CF1 29C0      <1>      sub     eax, eax
226      <1> ;sub     edx, edx
227 00006CF3 BF[E7380100] <1>      mov     edi, Current_Dir_Drv
228      <1>
229 00006CF8 A2[A8FA0000] <1>      mov     [Restore_CDIRE], al ; 0
230 00006CFD C3      <1>      retn
231      <1>
232      <1> dos_prompt:
233      <1>      ; 06/05/2016
234      <1>      ; 30/01/2016
235      <1>      ; 29/01/2016
236      <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
237      <1>      ; 15/09/2011
238      <1>      ; 13/09/2009
239      <1>      ; 2004-2005
240      <1>
241      <1>      ; 06/05/2016
242 00006CFE C705[A4450100]- <1>      mov     dword [mainprog_return_addr], return_from_cmd_interpreter
242 00006D04 [B26D0000] <1>
243      <1>
244      <1> loc_TRDOS_prompt:
245 00006D08 BF[E6390100] <1>      mov     edi, TextBuffer
246 00006D0D C6075B <1>      mov     byte [edi], "["
247 00006D10 47      <1>      inc     edi
248 00006D11 BE[FBFA0000] <1>      mov     esi, TRDOSPromptLabel
249      <1> get_next_prompt_label_char:
250 00006D16 803E20 <1>      cmp     byte [esi], 20h
251 00006D19 7203 <1>      jb      short pass_prompt_label
252 00006D1B A4      <1>      movsb
253 00006D1C EBF8 <1>      jmp     short get_next_prompt_label_char
254      <1> pass_prompt_label:
255 00006D1E C6075D <1>      mov     byte [edi], "]"
256 00006D21 47 <1>      inc     edi
257 00006D22 C60720 <1>      mov     byte [edi], 20h
258 00006D25 47 <1>      inc     edi
259 00006D26 BE[E7380100] <1>      mov     esi, Current_Dir_Drv
260 00006D2B 66A5 <1>      movsw
261 00006D2D A4 <1>      movsb
262      <1> loc_prompt_current_directory:
263 00006D2E 803E20 <1>      cmp     byte [esi], 20h

```

```

264 00006D31 7203      <1>      jb      short pass_prompt_current_directory
265 00006D33 A4        <1>      movsb
266 00006D34 EBF8      <1>      jmp      short loc_prompt_current_directory
267                                <1> pass_prompt_current_directory:
268 00006D36 C6073E    <1>      mov      byte [edi], '>'
269 00006D39 47        <1>      inc      edi
270 00006D3A C60700    <1>      mov      byte [edi], 0
271 00006D3D BE[E6390100] <1>      mov      esi, TextBuffer
272 00006D42 E876F5FFFF    <1>      call     print_msg
273                                <1>
274                                <1>      ;sub     bh, bh ; video page = 0
275                                <1>      ;call    get_cpos ; get cursor position
276 00006D47 668B15[3E380100] <1>      mov      dx, [CURSOR_POSN] ; video page 0
277 00006D4E 8815[46390100] <1>      mov      [CursorColumn], dl
278                                <1>
279                                <1>      ; 30/01/2016 (to show cursor on the row, again)
280                                <1>      ; (Initial color attributes of video page 0 is 0)
281                                <1>      ; (see: 'StartPMP' in trdos386.s)
282                                <1>      ;
283                                <1>      ;mov     edi, 0B8000h ; start of video page 0
284                                <1>      ;movzx   ecx, dl ; column
285                                <1>      ;mov     al, 80
286                                <1>      ;mul     dh
287                                <1>      ;add     ax, cx
288                                <1>      ;shl     ax, 1 ; character + attribute
289                                <1>      ;add     di, ax ; (2*80*row) + (2*column)
290                                <1>      ;neg     cl
291                                <1>      ;add     cl, 80
292                                <1>      ;mov     ax, 700h ; ah = 7 (color attribute)
293                                <1>      ;rep     stosw
294                                <1>
295                                <1> loc_rw_char:
296 00006D54 E899000000    <1>      call     rw_char
297                                <1> loc_move_command:
298 00006D59 BE[96390100] <1>      mov      esi, CommandBuffer
299 00006D5E 89F7        <1>      mov      edi, esi
300 00006D60 31C9        <1>      xor      ecx, ecx
301                                <1> first_command_char:
302 00006D62 AC          <1>      lodsb
303 00006D63 3C20        <1>      cmp      al, 20h
304 00006D65 772E        <1>      ja       short pass_space_control
305 00006D67 7241        <1>      jb       short loc_move_cmd_arguments_ok
306 00006D69 81FE[E5390100] <1>      cmp      esi, CommandBuffer + 79
307 00006D6F 72F1        <1>      jb       short first_command_char
308 00006D71 EB37        <1>      jmp      short loc_move_cmd_arguments_ok
309                                <1>
310                                <1> next_command_char:
311 00006D73 AC          <1>      lodsb
312 00006D74 3C20        <1>      cmp      al, 20h
313 00006D76 771D        <1>      ja       short pass_space_control
314 00006D78 7230        <1>      jb       short loc_move_cmd_arguments_ok
315                                <1>
316                                <1> loc_1st_cmd_arg: ; 30/01/2016
317 00006D7A AC          <1>      lodsb
318 00006D7B 3C20        <1>      cmp      al, 20h
319 00006D7D 74FB        <1>      je       short loc_1st_cmd_arg
320 00006D7F 7229        <1>      jb       short loc_move_cmd_arguments_ok
321                                <1>
322 00006D81 C60700    <1>      mov      byte [edi], 0
323 00006D84 47        <1>      inc      edi
324                                <1>
325                                <1> loc_move_cmd_arguments:
326 00006D85 AA          <1>      stosb
327 00006D86 81FE[E5390100] <1>      cmp      esi, CommandBuffer + 79
328 00006D8C 731C        <1>      jnb      short loc_move_cmd_arguments_ok
329 00006D8E AC          <1>      lodsb
330 00006D8F 3C20        <1>      cmp      al, 20h
331 00006D91 73F2        <1>      jnb      short loc_move_cmd_arguments
332 00006D93 EB15        <1>      jmp      short loc_move_cmd_arguments_ok
333                                <1>
334                                <1> pass_space_control:
335 00006D95 3C61        <1>      cmp      al, 61h
336 00006D97 7206        <1>      jb       short pass_capitalize
337 00006D99 3C7A        <1>      cmp      al, 7Ah
338 00006D9B 7702        <1>      ja       short pass_capitalize
339 00006D9D 24DF        <1>      and      al, 0DFh
340                                <1> pass_capitalize:
341 00006D9F AA          <1>      stosb
342 00006DA0 FEC1        <1>      inc      cl
343 00006DA2 81FE[E5390100] <1>      cmp      esi, CommandBuffer + 79
344 00006DA8 72C9        <1>      jb       short next_command_char
345                                <1>
346                                <1> loc_move_cmd_arguments_ok:
347 00006DAA C60700    <1>      mov      byte [edi], 0
348                                <1>
349                                <1> call_command_interpreter:
350 00006DAD E8D4080000    <1>      call     command_interpreter
351                                <1>
352                                <1> return_from_cmd_interpreter:
353 00006DB2 B950000000    <1>      mov      ecx, 80
354                                <1>      ;mov     cx, 80
355 00006DB7 BF[96390100] <1>      mov      edi, CommandBuffer
356 00006DBC 30C0        <1>      xor      al, al
357 00006DBE 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 00006DC0 803D[225E0000]03 <1>      cmp      byte [CRT_MODE], 3 ; 80*25 color
363 00006DC7 741D        <1>      je       short pass_set_txt_mode
364                                <1>
365 00006DC9 E895A7FFFF    <1>      call     set_txt_mode ; set vide mode to 03h

```

```

366          <1>      ; 07/01/2017
367          <1>      xor     al, al
368          <1>
369          <1> loc_check_active_page:
370          <1>      ;xor     al, al
371 00006DD0 3805[4E380100] <1>      cmp     [ACTIVE_PAGE], al ; 0
372 00006DD6 0F842CFFFFFF <1>      je      loc_TRDOS_prompt
373          <1>      ; AL = 0 = video page 0
374 00006DDC E89BABFFFF <1>      call    set_active_page
375 00006DE1 E922FFFFFF <1>      jmp     loc_TRDOS_prompt ; infinitive loop
376          <1>
377          <1> pass_set_txt_mode:
378 00006DE6 BE[13070100] <1>      mov     esi, nextline
379 00006DEB E8CDF4FFFF <1>      call    print_msg
380 00006DF0 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 00006DF2 30E4 <1>      xor     ah, ah
396 00006DF4 E81D9EFFFF <1>      call    int16h
397 00006DF9 20C0 <1>      and     al, al
398 00006DFB 7434 <1>      jz      short loc_arrow
399 00006DFD 3CE0 <1>      cmp     al, 0E0h
400 00006DFF 7430 <1>      je      short loc_arrow
401 00006E01 3C08 <1>      cmp     al, 08h
402 00006E03 7544 <1>      jne     short char_return
403          <1> loc_back:
404 00006E05 3A15[46390100] <1>      cmp     dl, [CursorColumn]
405 00006E0B 76E5 <1>      jna     short readnextchar
406          <1> prev_column:
407 00006E0D FECA <1>      dec     dl
408          <1> set_cursor_pos:
409 00006E0F 6652 <1>      push    dx
410          <1>      ;xor     bh, bh ; 0 = video page 0
411          <1>      ; DH = Row, DL = Column
412 00006E11 E832AFFFFF <1>      call    _set_cpos ; 17/01/2016
413 00006E16 665A <1>      pop     dx
414          <1>      ;movzx ebx, dl
415 00006E18 88D3 <1>      mov     bl, dl
416 00006E1A 2A1D[46390100] <1>      sub     bl, [CursorColumn]
417 00006E20 B020 <1>      mov     al, 20h
418 00006E22 8883[96390100] <1>      mov     [CommandBuffer+ebx], al
419          <1>      ;sub     bh, bh ; video page 0
420          <1>      ;mov     cx, 1
421 00006E28 B307 <1>      mov     bl, 7 ; color attribute
422 00006E2A E80AAEFFFF <1>      call    _write_c_current ; 17/01/2016
423          <1>      ;mov     dx, [CURSOR_POSN]
424 00006E2F EBC1 <1>      jmp     short readnextchar
425          <1> loc_arrow:
426 00006E31 80FC4B <1>      cmp     ah, 4Bh
427 00006E34 74CF <1>      je      short loc_back
428 00006E36 80FC53 <1>      cmp     ah, 53h
429 00006E39 74CA <1>      je      short loc_back
430 00006E3B 80FC4D <1>      cmp     ah, 4Dh
431 00006E3E 75B2 <1>      jne     short readnextchar
432 00006E40 80FA4F <1>      cmp     dl, 79
433 00006E43 73AD <1>      jnb     short readnextchar
434 00006E45 FEC2 <1>      inc     dl
435 00006E47 EBC6 <1>      jmp     short set_cursor_pos
436          <1> char_return:
437 00006E49 0FB6DA <1>      movzx   ebx, dl
438 00006E4C 2A1D[46390100] <1>      sub     bl, [CursorColumn]
439 00006E52 3C20 <1>      cmp     al, 20h
440 00006E54 7220 <1>      jb      short loc_escape
441 00006E56 8883[96390100] <1>      mov     [CommandBuffer+ebx], al
442 00006E5C 80FA4F <1>      cmp     dl, 79
443 00006E5F 7391 <1>      jnb     short readnextchar
444 00006E61 66BB0700 <1>      mov     bx, 7 ; color attribute
445 00006E65 E848AEFFFF <1>      call    _write_tty
446 00006E6A 668B15[3E380100] <1>      mov     dx, [CURSOR_POSN] ; video page 0
447 00006E71 E97CFFFFFF <1>      jmp     readnextchar
448          <1> loc_escape:
449 00006E76 3C1B <1>      cmp     al, 1Bh
450 00006E78 7418 <1>      je      short rw_char_retn
451          <1>      ;
452 00006E7A 3C0D <1>      cmp     al, 0Dh ; CR
453 00006E7C 0F8570FFFFFF <1>      jne     readnextchar
454          <1>      ; 13/05/2016
455 00006E82 66BB0700 <1>      mov     bx, 7 ; attribute/color (bl)
456          <1>      ; video page 0 (bh=0)
457 00006E86 E827AEFFFF <1>      call    _write_tty
458          <1>      ;mov     bx, 7 ; attribute/color
459          <1>      ; video page 0 (bh=0)
460 00006E8B B00A <1>      mov     al, 0Ah ; LF
461 00006E8D E820AEFFFF <1>      call    _write_tty
462          <1> rw_char_retn:
463 00006E92 C3 <1>      retn
464          <1>
465          <1> show_date:
466          <1>      ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
467          <1>      ; 2004-2005

```



```

468 <1>
469 <1> ;mov ah, 04h
470 <1> ;call int1Ah
471 00006E93 E820EBFFFF <1> call RTC_40 ; GET RTC DATE
472 <1>
473 00006E98 88D0 <1> mov al, dl
474 00006E9A E86E9DFFFF <1> call bcd_to_ascii
475 00006E9F 66A3[E7FB0000] <1> mov [Day], ax
476 <1>
477 00006EA5 88F0 <1> mov al, dh
478 00006EA7 E8619DFFFF <1> call bcd_to_ascii
479 00006EAC 66A3[EAFB0000] <1> mov [Month], ax
480 <1>
481 00006EB2 88E8 <1> mov al, ch
482 00006EB4 E8549DFFFF <1> call bcd_to_ascii
483 00006EB9 66A3[EDFB0000] <1> mov [Century], ax
484 <1>
485 00006EBF 88C8 <1> mov al, cl
486 00006EC1 E8479DFFFF <1> call bcd_to_ascii
487 00006EC6 66A3[EFFB0000] <1> mov word [Year], ax
488 <1>
489 00006ECC BE[D7FB0000] <1> mov esi, Msg_Show_Date
490 00006ED1 E8E7F3FFFF <1> call print_msg
491 <1>
492 00006ED6 C3 <1> retn
493 <1>
494 <1> set_date:
495 <1> ; 13/05/2016
496 <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
497 <1> ; 2004-2005
498 <1>
499 00006ED7 BE[BBFB0000] <1> mov esi, Msg_Enter_Date
500 00006EDC E8DCF3FFFF <1> call print_msg
501 <1>
502 <1> loc_enter_day_1:
503 00006EE1 30E4 <1> xor ah, ah
504 00006EE3 E82E9DFFFF <1> call int16h
505 <1> ; AL = ASCII Code of the Character
506 00006EE8 3C0D <1> cmp al, 13
507 00006EEA 0F84B7010000 <1> je loc_set_date_retn
508 00006EF0 3C1B <1> cmp al, 27
509 00006EF2 0F84AF010000 <1> je loc_set_date_retn
510 00006EF8 A2[E7FB0000] <1> mov [Day], al
511 00006EFD 3C30 <1> cmp al, '0'
512 00006EFF 0F82AD010000 <1> jb loc_set_date_stc_0
513 00006F05 3C33 <1> cmp al, '3'
514 00006F07 0F87A5010000 <1> ja loc_set_date_stc_0
515 <1> ; 13/05/2016
516 <1> ;mov bx, 7 ; attribute/color (bl)
517 <1> ; video page 0 (bh)
518 00006F0D B307 <1> mov bl, 7
519 00006F0F E89EADFFFF <1> call _write_tty
520 <1> loc_enter_day_2:
521 00006F14 30E4 <1> xor ah, ah
522 00006F16 E8FB9CFFFF <1> call int16h
523 <1> ; AL = ASCII Code of the Character
524 00006F1B 3C1B <1> cmp al, 27
525 00006F1D 0F8484010000 <1> je loc_set_date_retn
526 00006F23 A2[E8FB0000] <1> mov [Day+1], al
527 00006F28 3C30 <1> cmp al, '0'
528 00006F2A 0F828C010000 <1> jb loc_set_date_stc_1
529 00006F30 3C39 <1> cmp al, '9'
530 00006F32 0F8784010000 <1> ja loc_set_date_stc_1
531 00006F38 803D[E7FB0000]33 <1> cmp byte [Day], '3'
532 00006F3F 7208 <1> jb short pass_set_day_31
533 00006F41 3C31 <1> cmp al, '1'
534 00006F43 0F8773010000 <1> ja loc_set_date_stc_1
535 <1> pass_set_day_31:
536 <1> ; 13/05/2016
537 <1> ;mov bx, 7 ; attribute/color (bl)
538 <1> ; video page 0 (bh)
539 00006F49 B307 <1> mov bl, 7
540 00006F4B E862ADFFFF <1> call _write_tty
541 <1> loc_enter_separator_1:
542 00006F50 28E4 <1> sub ah, ah ; 0
543 00006F52 E8BF9CFFFF <1> call int16h
544 <1> ; AL = ASCII Code of the Character
545 00006F57 3C1B <1> cmp al, 27
546 00006F59 0F8448010000 <1> je loc_set_date_retn
547 00006F5F 3C2D <1> cmp al, '-'
548 00006F61 7408 <1> je short pass_set_date_separator_1
549 00006F63 3C2F <1> cmp al, '/'
550 00006F65 0F856C010000 <1> jne loc_set_date_stc_2
551 <1> pass_set_date_separator_1:
552 <1> ; 13/05/2016
553 <1> ;mov bx, 7 ; attribute/color (bl)
554 <1> ; video page 0 (bh)
555 00006F6B B307 <1> mov bl, 7
556 00006F6D E840ADFFFF <1> call _write_tty
557 <1> loc_enter_month_1:
558 00006F72 30E4 <1> xor ah, ah ; 0
559 00006F74 E89D9CFFFF <1> call int16h
560 <1> ; AL = ASCII Code of the Character
561 00006F79 3C1B <1> cmp al, 27
562 00006F7B 0F8426010000 <1> je loc_set_date_retn
563 00006F81 A2[EAFB0000] <1> mov [Month], al
564 00006F86 3C30 <1> cmp al, '0'
565 00006F88 0F8264010000 <1> jb loc_set_date_stc_3
566 00006F8E 3C31 <1> cmp al, '1'
567 00006F90 0F875C010000 <1> ja loc_set_date_stc_3
568 <1> ; 13/05/2016
569 <1> ;mov bx, 7 ; attribute/color (bl)

```

```

570                                     <1>             ; video page 0 (bh)
571 00006F96 B307                       <1>             mov     bl, 7
572 00006F98 E815ADFFFF                 <1>             call    _write_tty
573                                     <1> loc_enter_month_2:
574 00006F9D 30E4                       <1>             xor     ah, ah
575 00006F9F E8729CFFFF                 <1>             call    int16h
576                                     <1>             ; AL = ASCII Code of the Character
577 00006FA4 3C1B                       <1>             cmp     al, 27
578 00006FA6 0F84FB000000               <1>             je      loc_set_date_retn
579 00006FAC A2[EBFB0000]               <1>             mov     [Month+1], al
580 00006FB1 3C30                       <1>             cmp     al, '0'
581 00006FB3 0F8254010000               <1>             jb      loc_set_date_stc_4
582 00006FB9 3C39                       <1>             cmp     al, '9'
583 00006FBB 0F874C010000               <1>             ja      loc_set_date_stc_4
584 00006FC1 803D[EAFB0000]31          <1>             cmp     byte [Month], '1'
585 00006FC8 7208                       <1>             jb      short pass_set_month_12
586 00006FCA 3C32                       <1>             cmp     al, '2'
587 00006FCC 0F873B010000               <1>             ja      loc_set_date_stc_4
588                                     <1> pass_set_month_12:
589                                     <1>             ; 13/05/2016
590                                     <1>             ;mov    bx, 7 ; attribute/color (bl)
591                                     <1>             ; video page 0 (bh)
592 00006FD2 B307                       <1>             mov     bl, 7
593 00006FD4 E8D9ACFFFF                 <1>             call    _write_tty
594                                     <1> loc_enter_separator_2:
595 00006FD9 28E4                       <1>             sub     ah, ah
596 00006FDB E8369CFFFF                 <1>             call    int16h
597                                     <1>             ; AL = ASCII Code of the Character
598 00006FE0 3C1B                       <1>             cmp     al, 27
599 00006FE2 0F84BF000000               <1>             je      loc_set_date_retn
600 00006FE8 3C2D                       <1>             cmp     al, '-'
601 00006FEA 7408                       <1>             je      short pass_set_date_separator_2
602 00006FEC 3C2F                       <1>             cmp     al, '/'
603 00006FEE 0F8534010000               <1>             jne     loc_set_date_stc_5
604                                     <1> pass_set_date_separator_2:
605                                     <1>             ; 13/05/2016
606                                     <1>             ;mov    bx, 7 ; attribute/color (bl)
607                                     <1>             ; video page 0 (bh)
608 00006FF4 B307                       <1>             mov     bl, 7
609 00006FF6 E8B7ACFFFF                 <1>             call    _write_tty
610                                     <1> loc_enter_year_1:
611 00006FFB 30E4                       <1>             xor     ah, ah
612 00006FFD E8149CFFFF                 <1>             call    int16h
613                                     <1>             ; AL = ASCII Code of the Character
614 00007002 3C1B                       <1>             cmp     al, 27
615 00007004 0F849D000000               <1>             je      loc_set_date_retn
616 0000700A A2[EFFB0000]               <1>             mov     [Year], al
617 0000700F 3C30                       <1>             cmp     al, '0'
618 00007011 0F822C010000               <1>             jb      loc_set_date_stc_6
619 00007017 3C39                       <1>             cmp     al, '9'
620 00007019 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 0000701F B307                       <1>             mov     bl, 7
625 00007021 E88CACFFFF                 <1>             call    _write_tty
626                                     <1> loc_enter_year_2:
627 00007026 30E4                       <1>             xor     ah, ah
628 00007028 E8E99BFFFF                 <1>             call    int16h
629                                     <1>             ; AL = ASCII Code of the Character
630 0000702D 3C1B                       <1>             cmp     al, 27
631 0000702F 7476                       <1>             je      short loc_set_date_retn
632 00007031 A2[F0FB0000]               <1>             mov     byte [Year+1], al
633 00007036 3C30                       <1>             cmp     al, '0'
634 00007038 0F8220010000               <1>             jb      loc_set_date_stc_7
635 0000703E 3C39                       <1>             cmp     al, '9'
636 00007040 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 00007046 B307                       <1>             mov     bl, 7
641 00007048 E865ACFFFF                 <1>             call    _write_tty
642                                     <1> loc_set_date_get_lchar_again:
643 0000704D 28E4                       <1>             sub     ah, ah ; 0
644 0000704F E8C29BFFFF                 <1>             call    int16h
645                                     <1>             ; AL = ASCII Code of the Character
646 00007054 3C0D                       <1>             cmp     al, 13 ; ENTER key
647 00007056 7412                       <1>             je      short loc_set_date_progress
648 00007058 3C1B                       <1>             cmp     al, 27 ; ESC key
649 0000705A 744B                       <1>             je      short loc_set_date_retn
650                                     <1>             ;
651 0000705C E82A010000               <1>             call    check_for_backspace
652 00007061 75EA                       <1>             jne     short loc_set_date_get_lchar_again
653                                     <1>
654                                     <1> loc_set_date_bs_8:
655 00007063 E811010000               <1>             call    write_backspace
656 00007068 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 0000706A E849E9FFFF                 <1>             call    RTC_40 ; GET RTC DATE
663                                     <1>             ; CH = century (in BCD)
664                                     <1>
665 0000706F 66A1[EFFB0000]           <1>             mov     ax, [Year]
666 00007075 662D3030                 <1>             sub     ax, '00'
667 00007079 C0E004                   <1>             shl     al, 4 ; * 16
668 0000707C 88C1                     <1>             mov     cl, al
669 0000707E 00E1                     <1>             add     cl, ah
670 00007080 66A1[EAFB0000]           <1>             mov     ax, [Month]
671 00007086 662D3030                 <1>             sub     ax, '00'

```

```

672 0000708A C0E004      <1>      shl     al, 4 ; * 16
673 0000708D 88C6      <1>      mov     dh, al
674 0000708F 00E6      <1>      add     dh, ah
675 00007091 66A1[E7FB0000]    <1>      mov     ax, [Day]
676 00007097 662D3030      <1>      sub     ax, '00'
677 0000709B C0E004      <1>      shl     al, 4 ; * 16
678 0000709E 88C2      <1>      mov     dl, al
679 000070A0 00E2      <1>      add     dl, ah
680                                <1>
681                                <1>      ;mov     ah, 05h
682                                <1>      ;call    int1Ah
683 000070A2 E83EE9FFFF    <1>      call    RTC_50 ; SET RTC DATE
684                                <1>
685                                <1> loc_set_date_retn:
686 000070A7 BE[13070100] <1>      mov     esi, nextline
687 000070AC E80CF2FFFF    <1>      call    print_msg
688 000070B1 C3      <1>      retn
689                                <1>
690                                <1> loc_set_date_stc_0:
691                                <1>      ;xor     bh, bh ; video page 0
692 000070B2 E8DBACFFFF    <1>      call    beeper ; BEEP !
693 000070B7 E925FEFFFF    <1>      jmp     loc_enter_day_1
694                                <1> loc_set_date_stc_1:
695 000070BC E8CA000000    <1>      call    check_for_backspace
696 000070C1 740A      <1>      je      short loc_set_date_bs_1
697                                <1>      ;xor     bh, bh ; video page 0
698 000070C3 E8CAACFFFF    <1>      call    beeper ; BEEP !
699 000070C8 E947FEFFFF    <1>      jmp     loc_enter_day_2
700                                <1> loc_set_date_bs_1:
701 000070CD E8A7000000    <1>      call    write_backspace
702 000070D2 E90AFEFFFF    <1>      jmp     loc_enter_day_1
703                                <1> loc_set_date_stc_2:
704 000070D7 E8AF000000    <1>      call    check_for_backspace
705 000070DC 740A      <1>      je      short loc_set_date_bs_2
706                                <1>      ;xor     bh, bh ; video page 0
707 000070DE E8AFACFFFF    <1>      call    beeper ; BEEP !
708 000070E3 E968FEFFFF    <1>      jmp     loc_enter_separator_1
709                                <1> loc_set_date_bs_2:
710 000070E8 E88C000000    <1>      call    write_backspace
711 000070ED E922FEFFFF    <1>      jmp     loc_enter_day_2
712                                <1> loc_set_date_stc_3:
713 000070F2 E894000000    <1>      call    check_for_backspace
714 000070F7 740A      <1>      je      short loc_set_date_bs_3
715                                <1>      ;xor     bh, bh ; video page 0
716 000070F9 E894ACFFFF    <1>      call    beeper ; BEEP !
717 000070FE E96FFEFFFF    <1>      jmp     loc_enter_month_1
718                                <1> loc_set_date_bs_3:
719 00007103 E871000000    <1>      call    write_backspace
720 00007108 E943FEFFFF    <1>      jmp     loc_enter_separator_1
721                                <1> loc_set_date_stc_4:
722 0000710D E879000000    <1>      call    check_for_backspace
723 00007112 740A      <1>      je      short loc_set_date_bs_4
724                                <1>      ;xor     bh, bh ; video page 0
725 00007114 E879ACFFFF    <1>      call    beeper ; BEEP !
726 00007119 E97FFEFFFF    <1>      jmp     loc_enter_month_2
727                                <1> loc_set_date_bs_4:
728 0000711E E856000000    <1>      call    write_backspace
729 00007123 E94AFEFFFF    <1>      jmp     loc_enter_month_1
730                                <1> loc_set_date_stc_5:
731 00007128 E85E000000    <1>      call    check_for_backspace
732 0000712D 740A      <1>      je      short loc_set_date_bs_5
733                                <1>      ;xor     bh, bh ; video page 0
734 0000712F E85EACFFFF    <1>      call    beeper ; BEEP !
735 00007134 E9A0FEFFFF    <1>      jmp     loc_enter_separator_2
736                                <1> loc_set_date_bs_5:
737 00007139 E83B000000    <1>      call    write_backspace
738 0000713E E95AFEFFFF    <1>      jmp     loc_enter_month_2
739                                <1> loc_set_date_stc_6:
740 00007143 E843000000    <1>      call    check_for_backspace
741 00007148 740A      <1>      je      short loc_set_date_bs_6
742                                <1>      ;xor     bh, bh ; video page 0
743 0000714A E843ACFFFF    <1>      call    beeper ; BEEP !
744 0000714F E9A7FEFFFF    <1>      jmp     loc_enter_year_1
745                                <1> loc_set_date_bs_6:
746 00007154 E820000000    <1>      call    write_backspace
747 00007159 E97BFEFFFF    <1>      jmp     loc_enter_separator_2
748                                <1> loc_set_date_stc_7:
749 0000715E E828000000    <1>      call    check_for_backspace
750 00007163 740A      <1>      je      short loc_set_date_bs_7
751                                <1>      ;xor     bh, bh ; video page 0
752 00007165 E828ACFFFF    <1>      call    beeper ; BEEP !
753 0000716A E9B7FEFFFF    <1>      jmp     loc_enter_year_2
754                                <1> loc_set_date_bs_7:
755 0000716F E805000000    <1>      call    write_backspace
756 00007174 E982FEFFFF    <1>      jmp     loc_enter_year_1
757                                <1>
758                                <1> write_backspace:
759                                <1>      ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
760 00007179 B008      <1>      mov     al, 08h ; BACKSPACE
761                                <1>      ; 13/05/2016
762 0000717B 66BB0700    <1>      mov     bx, 7 ; bl = attribute/color
763                                <1>      ; bh = video page = 0
764 0000717F E82EABFFFF    <1>      call    _write_tty
765 00007184 B020      <1>      mov     al, 20h ; BLANK/SPACE char
766                                <1>      ;mov     bx, 7 ; attribute/color
767                                <1>      ;call    _write_c_current
768                                <1>      ;retn
769 00007186 E9AEAAFFFF    <1>      jmp     _write_c_current
770                                <1>
771                                <1> check_for_backspace:
772                                <1>      ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
773 0000718B 663D080E    <1>      cmp     ax, 0E08h

```

```

774 0000718F 7410      <1>      je      short cfbs_retn
775 00007191 663DE04B   <1>      cmp     ax, 4BE0h
776 00007195 740A      <1>      je      short cfbs_retn
777 00007197 663D004B   <1>      cmp     ax, 4B00h
778 0000719B 7404      <1>      je      short cfbs_retn
779 0000719D 663DE053   <1>      cmp     ax, 53E0h
780                                <1> cfbs_retn:
781 000071A1 C3      <1>      retn
782                                <1>
783                                <1> show_time:
784                                <1>      ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
785                                <1>      ; 2004-2005
786                                <1>
787                                <1>      ;mov     ah, 02h
788                                <1>      ;call    int1Ah
789 000071A2 E8A0E7FFFF   <1>      call    RTC_20 ; GET RTC TIME
790                                <1>
791 000071A7 88E8      <1>      mov     al, ch
792 000071A9 E85F9AFFFF   <1>      call    bcd_to_ascii
793 000071AE 66A3[15FC0000] <1>      mov     [Hour], ax
794                                <1>
795 000071B4 88C8      <1>      mov     al, cl
796 000071B6 E8529AFFFF   <1>      call    bcd_to_ascii
797 000071BB 66A3[18FC0000] <1>      mov     [Minute], ax
798                                <1>
799 000071C1 88F0      <1>      mov     al, dh
800 000071C3 E8459AFFFF   <1>      call    bcd_to_ascii
801 000071C8 66A3[1BFC0000] <1>      mov     [Second], ax
802                                <1>
803 000071CE BE[05FC0000] <1>      mov     esi, Msg_Show_Time
804 000071D3 E8E5F0FFFF   <1>      call    print_msg
805 000071D8 C3      <1>      retn
806                                <1>
807                                <1> set_time:
808                                <1>      ; 13/05/2016
809                                <1>      ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
810                                <1>      ; 2004-2005
811                                <1>
812 000071D9 BE[F4FB0000] <1>      mov     esi, Msg_Enter_Time
813 000071DE E8DAF0FFFF   <1>      call    print_msg
814                                <1>
815                                <1> loc_enter_hour_1:
816 000071E3 30E4      <1>      xor     ah, ah
817 000071E5 E82C9AFFFF   <1>      call    int16h
818                                <1>      ; AL = ASCII Code of the Character
819 000071EA 3C0D      <1>      cmp     al, 13 ; ENTER key
820 000071EC 0F84AE010000 <1>      je      loc_set_time_retn
821 000071F2 3C1B      <1>      cmp     al, 27 ; ESC key
822 000071F4 0F84A6010000 <1>      je      loc_set_time_retn
823 000071FA A2[15FC0000] <1>      mov     [Hour], al
824 000071FF 3C30      <1>      cmp     al, '0'
825 00007201 0F82A4010000 <1>      jnb     loc_set_time_stc_0
826 00007207 3C32      <1>      cmp     al, '2'
827 00007209 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 0000720F B307      <1>      mov     bl, 7
832 00007211 E89CAAFFFF   <1>      call    _write_tty
833                                <1> loc_enter_hour_2:
834 00007216 30E4      <1>      xor     ah, ah
835 00007218 E8F999FFFF   <1>      call    int16h
836                                <1>      ; AL = ASCII Code of the Character
837 0000721D 3C1B      <1>      cmp     al, 27
838 0000721F 0F847B010000 <1>      je      loc_set_time_retn
839 00007225 A2[16FC0000] <1>      mov     [Hour+1], al
840 0000722A 3C30      <1>      cmp     al, '0'
841 0000722C 0F8283010000 <1>      jnb     loc_set_time_stc_1
842 00007232 3C39      <1>      cmp     al, '9'
843 00007234 0F877B010000 <1>      ja      loc_set_time_stc_1
844 0000723A 803D[15FC0000]32 <1>      cmp     byte [Hour], '2'
845 00007241 7208      <1>      jnb     short pass_set_time_24
846 00007243 3C34      <1>      cmp     al, '4'
847 00007245 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 0000724B B307      <1>      mov     bl, 7
853 0000724D E860AAFFFF   <1>      call    _write_tty
854                                <1> loc_enter_time_separator_1:
855 00007252 28E4      <1>      sub     ah, ah ; 0
856 00007254 E8BD99FFFF   <1>      call    int16h
857                                <1>      ; AL = ASCII Code of the Character
858 00007259 3C1B      <1>      cmp     al, 27
859 0000725B 0F843F010000 <1>      je      loc_set_time_retn
860 00007261 3C3A      <1>      cmp     al, ':'
861 00007263 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 00007269 B307      <1>      mov     bl, 7
866 0000726B E842AAFFFF   <1>      call    _write_tty
867                                <1> loc_enter_minute_1:
868 00007270 30E4      <1>      xor     ah, ah
869 00007272 E89F99FFFF   <1>      call    int16h
870                                <1>      ; AL = ASCII Code of the Character
871 00007277 3C1B      <1>      cmp     al, 27
872 00007279 0F8421010000 <1>      je      loc_set_time_retn
873 0000727F A2[18FC0000] <1>      mov     [Minute], al
874 00007284 3C30      <1>      cmp     al, '0'
875 00007286 0F825F010000 <1>      jnb     loc_set_time_stc_3

```



```

876 0000728C 3C35      <1>      cmp     al, '5'
877 0000728E 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 00007294 B307      <1>      mov     bl, 7
882 00007296 E817AAFFFF <1>      call    _write_tty
883                                <1> loc_enter_minute_2:
884 0000729B 30E4      <1>      xor     ah, ah
885 0000729D E87499FFFF <1>      call    int16h
886                                <1>      ; AL = ASCII Code of the Character
887 000072A2 3C1B      <1>      cmp     al, 27
888 000072A4 0F84F6000000 <1>      je      loc_set_time_retn
889 000072AA A2[19FC0000] <1>      mov     [Minute+1], al
890 000072AF 3C30      <1>      cmp     al, '0'
891 000072B1 0F824F010000 <1>      jb      loc_set_time_stc_4
892 000072B7 3C39      <1>      cmp     al, '9'
893 000072B9 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 000072BF B307      <1>      mov     bl, 7
898 000072C1 E8ECA9FFFF <1>      call    _write_tty
899                                <1> loc_enter_time_separator_2:
900 000072C6 66C705[1BFC0000]30- <1>      mov     word [Second], 3030h
900 000072CE 30      <1>
901 000072CF 28E4      <1>      sub     ah, ah
902 000072D1 E84099FFFF <1>      call    int16h
903                                <1>      ; AL = ASCII Code of the Character
904 000072D6 3C0D      <1>      cmp     al, 13
905 000072D8 0F8485000000 <1>      je      loc_set_time_progress
906 000072DE 3C1B      <1>      cmp     al, 27
907 000072E0 0F84BA000000 <1>      je      loc_set_time_retn
908 000072E6 3C3A      <1>      cmp     al, ':'
909 000072E8 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 000072EE B307      <1>      mov     bl, 7
914 000072F0 E8BDA9FFFF <1>      call    _write_tty
915                                <1> loc_enter_second_1:
916 000072F5 30E4      <1>      xor     ah, ah
917 000072F7 E81A99FFFF <1>      call    int16h
918                                <1>      ; AL = ASCII Code of the Character
919 000072FC 3C0D      <1>      cmp     al, 13
920 000072FE 7463      <1>      je      short loc_set_time_progress
921 00007300 3C1B      <1>      cmp     al, 27
922 00007302 0F8498000000 <1>      je      loc_set_time_retn
923 00007308 A2[1BFC0000] <1>      mov     [Second], al
924 0000730D 3C30      <1>      cmp     al, '0'
925 0000730F 0F8227010000 <1>      jb      loc_set_time_stc_6
926 00007315 3C35      <1>      cmp     al, '5'
927 00007317 0F871F010000 <1>      ja      loc_set_time_stc_6
928                                <1>      ; 13/05/2016
929                                <1>      ;mov    bx, 7 ; attribute/color (bl)
930                                <1>      ; video page 0 (bh)
931 0000731D B307      <1>      mov     bl, 7
932 0000731F E88EA9FFFF <1>      call    _write_tty
933                                <1> loc_enter_second_2:
934 00007324 30E4      <1>      xor     ah, ah
935 00007326 E8EB98FFFF <1>      call    int16h
936                                <1>      ; AL = ASCII Code of the Character
937 0000732B 3C1B      <1>      cmp     al, 27
938 0000732D 7471      <1>      je      short loc_set_time_retn
939 0000732F 3C30      <1>      cmp     al, '0'
940 00007331 0F8229010000 <1>      jb      loc_set_time_stc_7
941 00007337 3C39      <1>      cmp     al, '9'
942 00007339 0F8721010000 <1>      ja      loc_set_time_stc_7
943                                <1>      ; 13/05/2016
944                                <1>      ;mov    bx, 7 ; attribute/color (bl)
945                                <1>      ; video page 0 (bh)
946 0000733F B307      <1>      mov     bl, 7
947 00007341 E86CA9FFFF <1>      call    _write_tty
948                                <1> loc_set_time_get_lchar_again:
949 00007346 28E4      <1>      sub     ah, ah ; 0
950 00007348 E8C998FFFF <1>      call    int16h
951                                <1>      ; AL = ASCII Code of the Character
952 0000734D 3C0D      <1>      cmp     al, 13
953 0000734F 7412      <1>      je      short loc_set_time_progress
954 00007351 3C1B      <1>      cmp     al, 27
955 00007353 744B      <1>      je      short loc_set_time_retn
956                                <1>      ;
957 00007355 E831FEFFFF <1>      call    check_for_backspace
958 0000735A 75EA      <1>      jne     short loc_set_time_get_lchar_again
959                                <1>
960                                <1> loc_set_time_bs_8:
961 0000735C E818FEFFFF <1>      call    write_backspace
962 00007361 EBC1      <1>      jmp     short loc_enter_second_2
963                                <1>
964                                <1> loc_set_time_progress:
965                                <1>      ; Get Current Time
966                                <1>      ;mov    ah, 02h
967                                <1>      ;call   int1Ah
968 00007363 E8DFE5FFFF <1>      call    RTC_20 ; GET RTC TIME
969                                <1>      ;DL = Daylight Savings Enable option (0-1)
970                                <1>
971 00007368 66A1[15FC0000] <1>      mov     ax, [Hour]
972 0000736E 662D3030 <1>      sub     ax, '00'
973 00007372 C0E004 <1>      shl     al, 4 ; * 16
974 00007375 88C5 <1>      mov     ch, al
975 00007377 00E5 <1>      add     ch, ah
976 00007379 66A1[18FC0000] <1>      mov     ax, [Minute]

```

```

977 0000737F 662D3030      <1>      sub     ax, '00'
978 00007383 C0E004      <1>      shl     al, 4 ; * 16
979 00007386 88C1      <1>      mov     cl, al
980 00007388 00E1      <1>      add     cl, ah
981 0000738A 66A1[1BFC0000] <1>      mov     ax, [Second]
982 00007390 662D3030      <1>      sub     ax, '00'
983 00007394 C0E004      <1>      shl     al, 4 ; * 16
984 00007397 88C6      <1>      mov     dh, al
985 00007399 00E6      <1>      add     dh, ah
986                                <1>
987                                <1>      ;mov     ah, 03h
988                                <1>      ;call    int1Ah
989 0000739B E8D6E5FFFF      <1>      call    RTC_30 ; SET RTC TIME
990                                <1>
991                                <1> loc_set_time_retn:
992 000073A0 BE[13070100] <1>      mov     esi, nextline
993 000073A5 E813EFFFFF      <1>      call    print_msg
994 000073AA C3      <1>      retn
995                                <1>
996                                <1> loc_set_time_stc_0:
997                                <1>      ;xor     bh, bh ; video page 0
998 000073AB E8E2A9FFFF      <1>      call    beeper ; BEEP !
999 000073B0 E92EFEFFFF      <1>      jmp     loc_enter_hour_1
1000                                <1> loc_set_time_stc_1:
1001 000073B5 E8D1FDFFFF      <1>      call    check_for_backspace
1002 000073BA 740A      <1>      je      short loc_set_time_bs_1
1003                                <1>      ;xor     bh, bh ; video page 0
1004 000073BC E8D1A9FFFF      <1>      call    beeper ; BEEP !
1005 000073C1 E950FEFFFF      <1>      jmp     loc_enter_hour_2
1006                                <1> loc_set_time_bs_1:
1007 000073C6 E8AEFDFFFF      <1>      call    write_backspace
1008 000073CB E913FEFFFF      <1>      jmp     loc_enter_hour_1
1009                                <1> loc_set_time_stc_2:
1010 000073D0 E8B6FDFFFF      <1>      call    check_for_backspace
1011 000073D5 740A      <1>      je      short loc_set_time_bs_2
1012                                <1>      ;xor     bh, bh ; video page 0
1013 000073D7 E8B6A9FFFF      <1>      call    beeper ; BEEP !
1014 000073DC E971FEFFFF      <1>      jmp     loc_enter_time_separator_1
1015                                <1> loc_set_time_bs_2:
1016 000073E1 E893FDFFFF      <1>      call    write_backspace
1017 000073E6 E92BFEFFFF      <1>      jmp     loc_enter_hour_2
1018                                <1> loc_set_time_stc_3:
1019 000073EB E89BFDFFFF      <1>      call    check_for_backspace
1020 000073F0 740A      <1>      je      short loc_set_time_bs_3
1021                                <1>      ;xor     bh, bh ; video page 0
1022 000073F2 E89BA9FFFF      <1>      call    beeper ; BEEP !6
1023 000073F7 E974FEFFFF      <1>      jmp     loc_enter_minute_1
1024                                <1> loc_set_time_bs_3:
1025 000073FC E878FDFFFF      <1>      call    write_backspace
1026 00007401 E94CFEFFFF      <1>      jmp     loc_enter_time_separator_1
1027                                <1> loc_set_time_stc_4:
1028 00007406 E880FDFFFF      <1>      call    check_for_backspace
1029 0000740B 740A      <1>      je      short loc_set_time_bs_4
1030                                <1>      ;xor     bh, bh ; video page 0
1031 0000740D E880A9FFFF      <1>      call    beeper ; BEEP !
1032 00007412 E984FEFFFF      <1>      jmp     loc_enter_minute_2
1033                                <1> loc_set_time_bs_4:
1034 00007417 E85DFDFFFF      <1>      call    write_backspace
1035 0000741C E94FFEFFFF      <1>      jmp     loc_enter_minute_1
1036                                <1> loc_set_time_stc_5:
1037 00007421 E865FDFFFF      <1>      call    check_for_backspace
1038 00007426 740A      <1>      je      short loc_set_time_bs_5
1039                                <1>      ;xor     bh, bh ; video page 0
1040 00007428 E865A9FFFF      <1>      call    beeper ; BEEP !
1041 0000742D E994FEFFFF      <1>      jmp     loc_enter_time_separator_2
1042                                <1> loc_set_time_bs_5:
1043 00007432 E842FDFFFF      <1>      call    write_backspace
1044 00007437 E95FFEFFFF      <1>      jmp     loc_enter_minute_2
1045                                <1> loc_set_time_stc_6:
1046 0000743C E84AFDFFFF      <1>      call    check_for_backspace
1047 00007441 7413      <1>      je      short loc_set_time_bs_6
1048                                <1>      ;xor     bh, bh ; video page 0
1049 00007443 E84AA9FFFF      <1>      call    beeper ; BEEP !
1050 00007448 66C705[1BFC0000]30- <1>      mov     word [Second], 3030h
1050 00007450 30      <1>
1051 00007451 E99FFEFFFF      <1>      jmp     loc_enter_second_1
1052                                <1> loc_set_time_bs_6:
1053 00007456 E81EFDFFFF      <1>      call    write_backspace
1054 0000745B E966FEFFFF      <1>      jmp     loc_enter_time_separator_2
1055                                <1> loc_set_time_stc_7:
1056 00007460 E826FDFFFF      <1>      call    check_for_backspace
1057 00007465 740A      <1>      je      short loc_set_time_bs_7
1058                                <1>      ;xor     bh, bh ; video page 0
1059 00007467 E826A9FFFF      <1>      call    beeper ; BEEP !
1060 0000746C E9B3FEFFFF      <1>      jmp     loc_enter_second_2
1061                                <1> loc_set_time_bs_7:
1062 00007471 E803FDFFFF      <1>      call    write_backspace
1063 00007476 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      0000747B 88C4      <1>      mov     ah, al
1082      0000747D 28C0      <1>      sub     al, al
1083      0000747F 0FB7F0      <1>      movzx   esi, ax
1084      00007482 81C600010900 <1>      add     esi, Logical_DOSDisks
1085      00007488 8A06      <1>      mov     al, [esi]
1086      0000748A 3C41      <1>      cmp     al, 'A'
1087      0000748C 7304      <1>      jnb     short loc_pvi_set_vol_name
1088      0000748E 8A6604      <1>      mov     ah, [esi+LD_FSType]
1089      00007491 C3      <1>      retn
1090      <1>
1091      <1> loc_pvi_set_vol_name:
1092      00007492 A2[4FFC0000] <1>      mov     [Vol_Drv_Name], al
1093      00007497 56      <1>      push    esi
1094      00007498 E858010000 <1>      call   move_volume_name_and_serial_no ;;;
1095      0000749D 7302      <1>      jnc     short loc_pvi_mvn_ok
1096      0000749F 5E      <1>      pop     esi
1097      000074A0 C3      <1>      retn
1098      <1>
1099      <1> loc_pvi_mvn_ok:
1100      000074A1 8B3424      <1>      mov     esi, [esp]
1101      000074A4 807E04A1 <1>      cmp     byte [esi+LD_FSType], 0A1h
1102      000074A8 7509      <1>      jne     short loc_pvi_fat_vol_size
1103      000074AA 8B4670      <1>      mov     eax, [esi+LD_FS_VolumeSize]
1104      000074AD 0FB75E11 <1>      movzx   ebx, word [esi+LD_FS_BytesPerSec]
1105      000074B1 EB07      <1>      jmp     short loc_vol_size_mul32
1106      <1> loc_pvi_fat_vol_size:
1107      000074B3 8B4670      <1>      mov     eax, [esi+LD_TotalSectors]
1108      000074B6 0FB75E11 <1>      movzx   ebx, word [esi+LD_BPB+BPB_BytsPerSec]
1109      <1> loc_vol_size_mul32:
1110      000074BA F7E3      <1>      mul     ebx
1111      000074BC 09D2      <1>      or      edx, edx
1112      000074BE 7507      <1>      jnz     short loc_vol_size_in_kbytes
1113      <1> loc_vol_size_in_bytes:
1114      000074C0 B9[2DFC0000] <1>      mov     ecx, VolSize_Bytes
1115      000074C5 EB0D      <1>      jmp     short loc_write_vol_size_str
1116      <1> loc_vol_size_in_kbytes:
1117      000074C7 66BB0004 <1>      mov     bx, 1024
1118      000074CB F7F3      <1>      div     ebx
1119      000074CD B9[20FC0000] <1>      mov     ecx, VolSize_KiloBytes
1120      000074D2 31D2      <1>      xor     edx, edx ; 0
1121      <1> loc_write_vol_size_str:
1122      000074D4 890D[1F410100] <1>      mov     [VolSize_Unit1], ecx
1123      <1>      ;
1124      000074DA BF[35410100] <1>      mov     edi, Vol_Tot_Sec_Str_End
1125      <1>      ;mov byte [edi], 0
1126      000074DF B90A000000 <1>      mov     ecx, 10
1127      <1> loc_write_vol_size_chr:
1128      000074E4 F7F1      <1>      div     ecx
1129      000074E6 80C230 <1>      add     dl, '0'
1130      000074E9 4F      <1>      dec     edi
1131      000074EA 8817      <1>      mov     [edi], dl
1132      000074EC 85C0      <1>      test    eax, eax
1133      000074EE 7404      <1>      jz      short loc_write_vol_size_str_ok
1134      000074F0 28D2      <1>      sub     dl, dl ; 0
1135      000074F2 EBF0      <1>      jmp     short loc_write_vol_size_chr
1136      <1>
1137      <1> loc_write_vol_size_str_ok:
1138      000074F4 893D[27410100] <1>      mov     [Vol_Tot_Sec_Str_Start], edi
1139      <1>      ;
1140      000074FA BF[38FC0000] <1>      mov     edi, Vol_FS_Name
1141      000074FF 8A4E03 <1>      mov     cl, [esi+LD_FATType]
1142      00007502 20C9      <1>      and     cl, cl ; 0 ?
1143      00007504 7515      <1>      jnz     short loc_write_vol_FAT_str_1
1144      00007506 66C7075452 <1>      mov     word [edi], 'TR'
1145      0000750B C7470420465331 <1>      mov     dword [edi+4], ' FS1'
1146      <1>      ;movzx ebx, word [esi+LD_FS_BytesPerSec]
1147      00007512 668B5E11 <1>      mov     bx, [esi+LD_FS_BytesPerSec]
1148      00007516 8B4674 <1>      mov     eax, [esi+LD_FS_FreeSectors]
1149      00007519 EB36      <1>      jmp     short loc_vol_freespace_mul32
1150      <1>
1151      <1> loc_write_vol_FAT_str_1:
1152      0000751B 66B83332 <1>      mov     ax, '32' ; FAT32
1153      0000751F 80F902 <1>      cmp     cl, 2 ; [esi+LD_FATType]
1154      00007522 7708      <1>      ja      short loc_write_vol_FAT_str_2
1155      00007524 66B83132 <1>      mov     ax, '12' ; FAT12
1156      00007528 7202      <1>      jb      short loc_write_vol_FAT_str_2
1157      0000752A B436      <1>      mov     ah, '6' ; FAT16
1158      <1> loc_write_vol_FAT_str_2:
1159      0000752C C70746415420 <1>      mov     dword [edi], 'FAT '
1160      00007532 66894704 <1>      mov     word [edi+4], ax
1161      <1>      ;
1162      <1>      ;movzx ebx, word [esi+LD_BPB+BPB_BytsPerSec]
1163      00007536 668B5E11 <1>      mov     bx, [esi+LD_BPB+BPB_BytsPerSec]
1164      0000753A 8B4674 <1>      mov     eax, [esi+LD_FreeSectors]
1165      <1>
1166      <1> loc_vol_freespace_recalc0:
1167      <1>      ; 01/03/2016
1168      0000753D 83F8FF <1>      cmp     eax, 0FFFFFFFh
1169      00007540 720F <1>      jb      short loc_vol_freespace_mul32
1170      <1>      ;inc     eax ; 0
1171      00007542 20C9 <1>      and     cl, cl ; byte [esi+LD_FATType]
1172      00007544 740B <1>      jz      short loc_vol_freespace_mul32
1173      00007546 53 <1>      push    ebx
1174      00007547 66BB00FF <1>      mov     bx, 0FF00h ; recalculate free sectors
1175      0000754B E8A9490000 <1>      call   calculate_fat_freespace
1176      00007550 5B <1>      pop     ebx
1177      <1>
1178      <1> loc_vol_freespace_mul32:
1179      00007551 F7E3 <1>      mul     ebx

```

```

1180 00007553 09D2      <1>      or      edx, edx
1181 00007555 7507      <1>      jnz     short loc_vol_fspace_in_kbytes
1182                      <1> loc_vol_fspace_in_bytes:
1183 00007557 B9[2DFC0000]    <1>      mov     ecx, VolSize_Bytes
1184 0000755C EB0D      <1>      jmp     short loc_write_vol_fspace_str
1185                      <1> loc_vol_fspace_in_kbytes:
1186 0000755E 66BB0004    <1>      mov     bx, 1024
1187 00007562 F7F3      <1>      div     ebx
1188 00007564 B9[20FC0000]    <1>      mov     ecx, VolSize_KiloBytes
1189 00007569 31D2      <1>      xor     edx, edx ; 0
1190                      <1> loc_write_vol_fspace_str:
1191 0000756B 890D[23410100] <1>      mov     [VolSize_Unit2], ecx
1192                      <1>      ;
1193 00007571 BF[45410100]    <1>      mov     edi, Vol_Free_Sectors_Str_End
1194                      <1>      ;mov byte [edi], 0
1195 00007576 B90A000000    <1>      mov     ecx, 10
1196                      <1> loc_write_vol_fspace_chr:
1197 0000757B F7F1      <1>      div     ecx
1198 0000757D 80C230    <1>      add     dl, '0'
1199 00007580 4F        <1>      dec     edi
1200 00007581 8817      <1>      mov     [edi], dl
1201 00007583 85C0      <1>      test    eax, eax
1202 00007585 7404      <1>      jz      short loc_write_vol_fspace_str_ok
1203 00007587 28D2      <1>      sub     dl, dl ; 0
1204 00007589 EBF0      <1>      jmp     short loc_write_vol_fspace_chr
1205                      <1>
1206                      <1> loc_write_vol_fspace_str_ok:
1207 0000758B 893D[37410100] <1>      mov     [Vol_Free_Sectors_Str_Start], edi
1208                      <1>      ;
1209 00007591 BE[36FC0000] <1>      mov     esi, Volume_in_drive
1210 00007596 E822EDFFFF    <1>      call    print_msg
1211 0000759B BE[76FC0000] <1>      mov     esi, Vol_Name
1212 000075A0 E818EDFFFF    <1>      call    print_msg
1213 000075A5 BE[13070100] <1>      mov     esi, nextline
1214 000075AA E80EEDFFFF    <1>      call    print_msg
1215                      <1>      ;
1216 000075AF BE[D7FC0000] <1>      mov     esi, Vol_Total_Sector_Header
1217 000075B4 E804EDFFFF    <1>      call    print_msg
1218 000075B9 8B35[27410100] <1>      mov     esi, [Vol_Tot_Sec_Str_Start]
1219 000075BF E8F9ECFFFF    <1>      call    print_msg
1220 000075C4 8B35[1F410100] <1>      mov     esi, [VolSize_Unit1]
1221 000075CA E8EEECFFFF    <1>      call    print_msg
1222                      <1>      ;
1223 000075CF BE[E8FC0000] <1>      mov     esi, Vol_Free_Sectors_Header
1224 000075D4 E8E4ECFFFF    <1>      call    print_msg
1225 000075D9 8B35[37410100] <1>      mov     esi, [Vol_Free_Sectors_Str_Start]
1226 000075DF E8D9ECFFFF    <1>      call    print_msg
1227 000075E4 8B35[23410100] <1>      mov     esi, [VolSize_Unit2]
1228 000075EA E8CEECFFFF    <1>      call    print_msg
1229                      <1>      ;
1230 000075EF 5E        <1>      pop     esi
1231                      <1>
1232                      <1>      ;mov ah, [esi+LD_FSType]
1233                      <1>      ;mov al, [esi+LD_FATType]
1234 000075F0 668B4603    <1>      mov     ax, [esi+LD_FATType]
1235                      <1>
1236 000075F4 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 000075F5 BF[76FC0000] <1>      mov     edi, Vol_Name
1254                      <1>
1255                      <1>      ;mov ah, [esi+LD_FSType]
1256                      <1>      ;mov al, [esi+LD_FATType]
1257 000075FA 668B4603    <1>      mov     ax, [esi+LD_FATType]
1258 000075FE 80FCA1    <1>      cmp     ah, 0A1h
1259 00007601 7418      <1>      je      short mvn_2
1260 00007603 08E4      <1>      or      ah, ah
1261 00007605 7404      <1>      jz      short mvn_0
1262 00007607 08C0      <1>      or      al, al
1263 00007609 7504      <1>      jnz     short mvn_1
1264                      <1> mvn_0:
1265 0000760B 8A06      <1>      mov     al, [esi]
1266 0000760D F9        <1>      stc
1267 0000760E C3        <1>      retn
1268                      <1> mvn_1:
1269 0000760F 3C02      <1>      cmp     al, 2
1270 00007611 7717      <1>      ja      short mvn_3
1271                      <1>      ;or al, al
1272                      <1>      ;jz short mvn_2
1273 00007613 8B462D    <1>      mov     eax, [esi+LD_BPB+VolumeID]
1274 00007616 83C631    <1>      add     esi, LD_BPB+VolumeLabel
1275 00007619 EB15      <1>      jmp     short mvn_4
1276                      <1> mvn_2:
1277 0000761B 8B4628    <1>      mov     eax, [esi+LD_FS_VolumeSerial]
1278 0000761E 83C62C    <1>      add     esi, LD_FS_VolumeName
1279 00007621 B910000000    <1>      mov     ecx, 16
1280 00007626 F3A5      <1>      rep     movsd
1281 00007628 EB10      <1>      jmp     short mvn_5

```



```

1282
1283 0000762A 8B4649
1284 0000762D 83C64D
1285
1286 00007630 B90B000000
1287 00007635 F3A4
1288 00007637 C60700
1289
1290
1291 0000763A E8CABCFFFF
1292 0000763F 8915[CBFC0000]
1293 00007645 A3[D0FC0000]
1294
1295 0000764A C3
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307 0000764B 31DB
1308 0000764D 88D7
1309 0000764F 3815[A7FA0000]
1310 00007655 7304
1311
1312 00007657 31C0
1313 00007659 F9
1314 0000765A C3
1315
1316 0000765B 56
1317 0000765C BE00010900
1318 00007661 01DE
1319 00007663 8A5E03
1320 00007666 20DB
1321 00007668 740F
1322 0000766A 80FB02
1323 0000766D 7705
1324
1325 0000766F 83C62D
1326 00007672 EB0E
1327
1328 00007674 83C649
1329 00007677 EB09
1330
1331 00007679 807E04A1
1332 0000767D 75D8
1333 0000767F 83C628
1334
1335 00007682 8B06
1336 00007684 5E
1337 00007685 C3
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359 00007686 C605[D8410100]00
1360 0000768D 80F904
1361 00007690 0F87AE020000
1362 00007696 0F822E010000
1363
1364
1365
1366 0000769C BF[15FB0000]
1367 000076A1 E8BA030000
1368 000076A6 7208
1369
1370 000076A8 C605[D8410100]01
1371 000076AF C3
1372
1373
1374 000076B0 B104
1375 000076B2 BF[31FB0000]
1376 000076B7 E8A4030000
1377 000076BC 720B
1378
1379 000076BE E8D0F7FFFF
1380 000076C3 E80FF8FFFF
1381 000076C8 C3
1382
1383

<1> mvn_3:
<1>     mov     eax, [esi+LD_BPB+FAT32_VolID]
<1>     add     esi, LD_BPB+FAT32_VolLab
<1> mvn_4:
<1>     mov     ecx, 11
<1>     rep     movsb
<1>     mov     byte [edi], 0
<1> mvn_5:
<1>     ;mov     [Current_VolSerial], eax
<1>     call    dwordtohex
<1>     mov     [Vol_Serial1], edx
<1>     mov     [Vol_Serial2], eax
<1>     ; ecx = 0
<1>     retn
<1>
<1> get_volume_serial_number:
<1>     ; 19/01/2016 (TRDOS 386 = TRDOS v2.0)
<1>     ; 08/08/2010
<1>     ;
<1>     ; INPUT -> DL = Logical DOS Drive number
<1>     ; OUTPUT -> EAX = Volume serial number
<1>     ;       BL= FAT Type
<1>     ;       BH = Logical DOS drv Number (DL input)
<1>     ; cf = 1 -> Drive not ready
<1>
<1>     xor     ebx, ebx
<1>     mov     bh, dl
<1>     cmp     [Last_DOS_DiskNo], dl
<1>     jnb     short loc_gvsn_start
<1> loc_gvsn_stc_retn:
<1>     xor     eax, eax
<1>     stc
<1>     retn
<1> loc_gvsn_start:
<1>     push    esi
<1>     mov     esi, Logical_DOSDisks
<1>     add     esi, ebx
<1>     mov     bl, [esi+LD_FATType]
<1>     and     bl, bl
<1>     jz      short loc_gvsn_fs
<1>     cmp     bl, 2
<1>     ja      short loc_gvsn_fat32
<1> loc_gvsn_fat:
<1>     add     esi, LD_BPB + VolumeID
<1>     jmp     short loc_gvsn_return
<1> loc_gvsn_fat32:
<1>     add     esi, LD_BPB + FAT32_VolID
<1>     jmp     short loc_gvsn_return
<1> loc_gvsn_fs:
<1>     cmp     byte [esi+LD_FSType], 0A1h
<1>     jne     short loc_gvsn_stc_retn
<1>     add     esi, LD_FS_VolumeSerial
<1> loc_gvsn_return:
<1>     mov     eax, [esi]
<1>     pop     esi
<1>     retn
<1>
<1> ; CMD_INTR.ASM [ TRDOS Command Interpreter Procedure ]
<1> ; 09/11/2011
<1> ; 29/01/2005
<1>
<1> command_interpreter:
<1>     ; 16/10/2016
<1>     ; 12/10/2016
<1>     ; 13/05/2016
<1>     ; 07/05/2016
<1>     ; 04/03/2016
<1>     ; 04/02/2016
<1>     ; 03/02/2016
<1>     ; 30/01/2016
<1>     ; 29/01/2016 (TRDOS 386 = TRDOS 2.0)
<1>     ; 15/09/2011
<1>     ; 29/01/2005
<1>
<1>     ; Input: ecx = command word length (CL)
<1>     ;       CommandBuffer = Command string offset
<1>
<1>     mov     byte [Program_Exit], 0
<1>     cmp     cl, 4
<1>     ja      c_6
<1>     jb      c_2
<1> c_4:
<1>
<1> cmp_cmd_exit:
<1>     mov     edi, Cmd_Exit
<1>     call    cmp_cmd
<1>     jc      short cmp_cmd_date
<1>
<1>     mov     byte [Program_Exit], 1
<1>     retn
<1>
<1> cmp_cmd_date:
<1>     mov     cl, 4
<1>     mov     edi, Cmd_Date
<1>     call    cmp_cmd
<1>     jc      short cmp_cmd_time
<1>
<1>     call    show_date
<1>     call    set_date
<1>     retn
<1>
<1> cmp_cmd_time:

```

```

1384 000076C9 B104          <1>      mov     cl, 4
1385 000076CB BF[36FB0000]   <1>      mov     edi, Cmd_Time
1386 000076D0 E88B030000    <1>      call    cmp_cmd
1387 000076D5 720B          <1>      jc      short cmp_cmd_show
1388                                <1>
1389 000076D7 E8C6FAFFFF    <1>      call    show_time
1390 000076DC E8F8FAFFFF    <1>      call    set_time
1391 000076E1 C3              <1>      retn
1392                                <1>
1393                                <1> cmp_cmd_show:
1394 000076E2 B104          <1>      mov     cl, 4
1395 000076E4 BF[47FB0000]   <1>      mov     edi, Cmd_Show
1396 000076E9 E872030000    <1>      call    cmp_cmd
1397 000076EE 0F83EF090000    <1>      jnc     show_file
1398                                <1>
1399                                <1> cmp_cmd_echo:
1400 000076F4 B104          <1>      mov     cl, 4
1401 000076F6 BF[83FB0000]   <1>      mov     edi, Cmd_Echo
1402 000076FB E860030000    <1>      call    cmp_cmd
1403 00007700 721B          <1>      jc      short cmp_cmd_copy
1404                                <1>
1405                                <1>      ; 14/04/2016
1406 00007702 56              <1>      push    esi
1407                                <1> cmd_echo_asciiz:
1408 00007703 46              <1>      inc     esi
1409 00007704 8A06          <1>      mov     al, [esi]
1410 00007706 3C20          <1>      cmp     al, 20h
1411 00007708 73F9          <1>      jnb     short cmd_echo_asciiz
1412 0000770A C60600        <1>      mov     byte [esi], 0
1413 0000770D 5E              <1>      pop     esi
1414 0000770E E8AAEBFFFF    <1>      call    print_msg
1415 00007713 BE[5C070100]   <1>      mov     esi, NextLine
1416                                <1>      ;call print_msg
1417                                <1>      ;retn
1418 00007718 E9A0EBFFFF    <1>      jmp     print_msg
1419                                <1>
1420                                <1> cmp_cmd_copy:
1421 0000771D B104          <1>      mov     cl, 4
1422 0000771F BF[6AFB0000]   <1>      mov     edi, Cmd_Copy
1423 00007724 E837030000    <1>      call    cmp_cmd
1424 00007729 0F8305180000    <1>      jnc     copy_file
1425                                <1>
1426                                <1> cmp_cmd_move:
1427 0000772F B104          <1>      mov     cl, 4
1428 00007731 BF[6FFB0000]   <1>      mov     edi, Cmd_Move
1429 00007736 E825030000    <1>      call    cmp_cmd
1430 0000773B 0F8399160000    <1>      jnc     move_file
1431                                <1>
1432                                <1> cmp_cmd_path:
1433 00007741 B104          <1>      mov     cl, 4
1434 00007743 BF[74FB0000]   <1>      mov     edi, Cmd_Path
1435 00007748 E813030000    <1>      call    cmp_cmd
1436 0000774D 0F83391A0000    <1>      jnc     set_get_path
1437                                <1>
1438                                <1> cmp_cmd_beep:
1439 00007753 B104          <1>      mov     cl, 4
1440 00007755 BF[A1FB0000]   <1>      mov     edi, Cmd_Beep
1441 0000775A E801030000    <1>      call    cmp_cmd
1442 0000775F 720B          <1>      jc      short cmp_cmd_find
1443                                <1>      ; 13/05/2016
1444 00007761 8A3D[4E380100] <1>      mov     bh, [ptty] ; [ACTIVE_PAGE]
1445 00007767 E926A6FFFF    <1>      jmp     beeper
1446                                <1>
1447                                <1> cmp_cmd_find:
1448 0000776C B104          <1>      mov     cl, 4
1449 0000776E BF[7EFB0000]   <1>      mov     edi, Cmd_Find
1450 00007773 E8E8020000    <1>      call    cmp_cmd
1451 00007778 0F82C5020000    <1>      jc      cmp_cmd_external
1452                                <1>
1453                                <1>      ;call find_and_list_files
1454 0000777E E9EA220000    <1>      jmp     find_and_list_files
1455                                <1>      ;retn
1456                                <1>
1457                                <1> c_1:
1458 00007783 AD              <1>      lodsd
1459                                <1> cmp_cmd_help:
1460 00007784 3C3F          <1>      cmp     al, '?'
1461 00007786 751D          <1>      jne     short cmp_cmd_remark
1462                                <1>
1463 00007788 BE[07FB0000]   <1>      mov     esi, Command_List
1464                                <1> cmd_help_next_w:
1465 0000778D E82BEBFFFF    <1>      call    print_msg
1466                                <1>
1467 00007792 803E20        <1>      cmp     byte [esi], 20h ; 0
1468 00007795 7232          <1>      jnb     short cmd_help_retn
1469                                <1>
1470 00007797 56              <1>      push    esi
1471 00007798 BE[13070100]   <1>      mov     esi, nextline
1472 0000779D E81BEBFFFF    <1>      call    print_msg
1473 000077A2 5E              <1>      pop     esi
1474 000077A3 EBE8          <1>      jmp     short cmd_help_next_w
1475                                <1>
1476                                <1> cmp_cmd_remark:
1477 000077A5 3C2A          <1>      cmp     al, '*'
1478 000077A7 0F8596020000    <1>      jne     cmp_cmd_external
1479 000077AD 46              <1>      inc     esi
1480 000077AE BF[48390100]   <1>      mov     edi, Remark
1481 000077B3 8A06          <1>      mov     al, [esi]
1482 000077B5 3C20          <1>      cmp     al, 20h
1483 000077B7 7707          <1>      ja      short cmd_remark_write
1484 000077B9 89FE          <1>      mov     esi, edi ; Remark
1485 000077BB E9FDEAFFFF    <1>      jmp     print_msg

```

```

1486
1487
1488 000077C0 AA
1489 000077C1 AC
1490 000077C2 3C20
1491 000077C4 73FA
1492 000077C6 C60700
1493
1494
1495
1496
1497 000077C9 C3
1498
1499
1500 000077CA 80F902
1501 000077CD 0F87B1000000
1502 000077D3 BE[96390100]
1503 000077D8 72A9
1504
1505
1506 000077DA 66AD
1507 000077DC 663D4344
1508 000077E0 7553
1509 000077E2 46
1510
1511 000077E3 668B06
1512 000077E6 3C20
1513 000077E8 76DF
1514
1515 000077EA 80FC3A
1516 000077ED 7504
1517 000077EF 46
1518 000077F0 46
1519 000077F1 EB4B
1520
1521
1522
1523
1524
1525
1526
1527
1528 000077F3 B4CD
1529
1530 000077F5 E858230000
1531 000077FA 0F8372220000
1532
1533
1534 00007800 3C03
1535 00007802 740C
1536
1537 00007804 3C0F
1538 00007806 745B
1539 00007808 3C11
1540 0000780A 7457
1541 0000780C 3C13
1542 0000780E 7468
1543
1544
1545 00007810 6650
1546 00007812 BE[AAFD0000]
1547 00007817 E8A1EAF000
1548 0000781C 6658
1549 0000781E 3A25[E4380100]
1550 00007824 0F8348220000
1551 0000782A 8825[E4380100]
1552 00007830 E93D220000
1553
1554
1555
1556 00007835 80FC3A
1557 00007838 0F8505020000
1558
1559
1560 0000783E 803E20
1561 00007841 0F8706020000
1562
1563 00007847 24DF
1564 00007849 2C41
1565 0000784B 0F82FC010000
1566
1567 00007851 3A05[A7FA0000]
1568 00007857 770A
1569
1570 00007859 88C2
1571 0000785B E85DF3FFFF
1572 00007860 7201
1573 00007862 C3
1574
1575
1576 00007863 BE[67FD0000]
1577 00007868 E850EAF000
1578
1579
1580 0000786D 8A15[E6380100]
1581
1582 00007873 E945F3FFFF
1583
1584
1585
1586 00007878 BE[48FD0000]
1587 0000787D E83BEAF000

<1>
<1> cmd_remark_write:
<1>     stosb
<1>     lodsb
<1>     cmp     al, 20h
<1>     jnb     short cmd_remark_write
<1>     mov     byte [edi], 0
<1>
<1> cmd_help_retn:
<1> cmd_remark_retn:
<1> cd_retn:
<1>     retn
<1>
<1> c_2:
<1>     cmp     cl, 2
<1>     ja      c_3
<1>     mov     esi, CommandBuffer
<1>     jnb     short c_1
<1>
<1> cmp_cmd_cd:
<1>     lodsw
<1>     cmp     ax, 'CD'
<1>     jne     short cmp_cmd_drive
<1>     inc     esi
<1> cd_0:
<1>     mov     ax, [esi]
<1>     cmp     al, 20h
<1>     jna     short cd_retn
<1>     ; 10/02/2016
<1>     cmp     ah, ':'
<1>     jne     short cd_1
<1>     inc     esi
<1>     inc     esi
<1>     jmp     short cd_2
<1>
<1> cd_1: ; change current directory
<1>     ; 29/11/2009
<1>     ; AH = CDh ; to separate 'CD' command from others
<1>     ; for restoring current directory
<1>     ; 0CDh sign is for saving cdir into
<1>     ; DOS drv description table cdir area
<1>
<1>     mov     ah, 0CDh ; mov byte [CD_COMMAND], 0CDh
<1>
<1>     call    change_current_directory
<1>     jnc     change_prompt_dir_string
<1>
<1> cd_error_messages:
<1>     cmp     al, 3
<1>     je      short cd_path_not_found
<1>     ; 16/10/2016 (15h -> 15)
<1>     cmp     al, 15 ; drive not ready error
<1>     je      short cd_drive_not_ready
<1>     cmp     al, 17 ; read error
<1>     je      short cd_drive_not_ready
<1>     cmp     al, 19 ; ; Bad directory/path name
<1>     je      short cd_command_failed
<1>
<1> cd_path_not_found:
<1>     push    ax
<1>     mov     esi, Msg_Dir_Not_Found
<1>     call    print_msg
<1>     pop     ax
<1>     cmp     ah, [Current_Dir_Level]
<1>     jnb     change_prompt_dir_string
<1>     mov     [Current_Dir_Level], ah
<1>     jmp     change_prompt_dir_string
<1>
<1> cmp_cmd_drive: ; change current drive
<1>     ; C:, D:, E: etc.
<1>     cmp     ah, ':'
<1>     jne     cmp_cmd_external
<1>
<1> cd_2: ; 'CD C:', 'CD D:' ...
<1>     cmp     byte [esi], 20h
<1>     ja      loc_cmd_failed
<1>
<1>     and     al, 0DFh
<1>     sub     al, 'A'
<1>     jc      loc_cmd_failed
<1>
<1>     cmp     al, [Last_DOS_DiskNo]
<1>     ja      short cd_drive_not_ready
<1>
<1>     mov     dl, al
<1>     call    change_current_drive
<1>     jc      short cd_drive_not_ready
<1>     retn
<1>
<1> cd_drive_not_ready:
<1>     mov     esi, Msg_Not_Ready_Read_Err
<1>     call    print_msg
<1>
<1> cd_fail_drive_restart:
<1>     mov     dl, [Current_Drv]
<1>     ;call    change_current_drive
<1>     jmp     change_current_drive
<1>     ;retn
<1>
<1> cd_command_failed:
<1>     mov     esi, Msg_Bad_Command
<1>     call    print_msg

```

```

1588 00007882 EBE9      <1>      jmp      short cd_fail_drive_restart
1589
1590      <1>      c_3:
1591      <1>      cmp_cmd_dir:
1592 00007884 BF[07FB0000] <1>      mov      edi, Cmd_Dir
1593 00007889 E8D2010000 <1>      call     cmp_cmd
1594 0000788E 0F8371020000 <1>      jnc      print_directory_list
1595      <1>
1596      <1>      cmp_cmd_cls:
1597 00007894 B103      <1>      mov      cl, 3
1598 00007896 BF[43FB0000] <1>      mov      edi, Cmd_Cls
1599 0000789B E8C0010000 <1>      call     cmp_cmd
1600 000078A0 0F832DEAFFFF <1>      jnc      clear_screen
1601      <1>
1602      <1>      cmp_cmd_ver:
1603 000078A6 B103      <1>      mov      cl, 3
1604 000078A8 BF[11FB0000] <1>      mov      edi, Cmd_Ver
1605 000078AD E8AE010000 <1>      call     cmp_cmd
1606 000078B2 720A      <1>      jc       short cmp_cmd_mem
1607      <1>
1608 000078B4 BE[AFFA0000] <1>      mov      esi, mainprog_Version
1609      <1>      ;call print_msg
1610 000078B9 E9FFE9FFFF <1>      jmp      print_msg
1611      <1>      ;retn
1612      <1>
1613      <1>      cmp_cmd_mem:
1614 000078BE B103      <1>      mov      cl, 3
1615 000078C0 BF[79FB0000] <1>      mov      edi, Cmd_Mem
1616 000078C5 E896010000 <1>      call     cmp_cmd
1617 000078CA 0F8384B9FFFF <1>      jnc      memory_info
1618      <1>
1619      <1>      cmp_cmd_del:
1620 000078D0 B103      <1>      mov      cl, 3
1621 000078D2 BF[4CFB0000] <1>      mov      edi, Cmd_Del
1622 000078D7 E884010000 <1>      call     cmp_cmd
1623 000078DC 0F832D0F0000 <1>      jnc      delete_file
1624      <1>
1625      <1>      cmp_cmd_set:
1626 000078E2 B103      <1>      mov      cl, 3
1627 000078E4 BF[3FFB0000] <1>      mov      edi, Cmd_Set
1628 000078E9 E872010000 <1>      call     cmp_cmd
1629 000078EE 0F8310180000 <1>      jnc      set_get_env
1630      <1>
1631      <1>      cmp_cmd_run:
1632 000078F4 B103      <1>      mov      cl, 3
1633 000078F6 BF[3BFB0000] <1>      mov      edi, Cmd_Run
1634 000078FB E860010000 <1>      call     cmp_cmd
1635      <1>      ; 07/05/2016
1636 00007900 0F823D010000 <1>      jc       cmp_cmd_external
1637 00007906 E9471E0000 <1>      jmp      load_and_execute_file
1638      <1>      c_5:
1639      <1>      cmp_cmd_mkdir:
1640 0000790B BF[64FB0000] <1>      mov      edi, Cmd_Mkdir
1641 00007910 E84B010000 <1>      call     cmp_cmd
1642 00007915 0F838C0A0000 <1>      jnc      make_directory
1643      <1>
1644      <1>      cmp_cmd_rmdir:
1645 0000791B B105      <1>      mov      cl, 5
1646 0000791D BF[5EFB0000] <1>      mov      edi, Cmd_Rmdir
1647 00007922 E839010000 <1>      call     cmp_cmd
1648 00007927 0F83990B0000 <1>      jnc      delete_directory
1649      <1>
1650      <1>      cmp_cmd_chdir:
1651 0000792D B105      <1>      mov      cl, 5
1652 0000792F BF[9BFB0000] <1>      mov      edi, Cmd_Chdir
1653 00007934 E827010000 <1>      call     cmp_cmd
1654 00007939 0F8204010000 <1>      jc       cmp_cmd_external
1655      <1>
1656 0000793F E99FFEFFFF <1>      jmp      cd_0
1657      <1>
1658      <1>      c_6:
1659 00007944 80F906      <1>      cmp      cl, 6
1660 00007947 0F87DF000000 <1>      ja       c_8
1661 0000794D 72BC      <1>      jnb      short c_5
1662      <1>      cmp_cmd_prompt:
1663 0000794F BF[1AFB0000] <1>      mov      edi, Cmd_Prompt
1664 00007954 E807010000 <1>      call     cmp_cmd
1665 00007959 722E      <1>      jc       short cmp_cmd_volume
1666      <1>      get_prompt_name_fchar:
1667 0000795B AC      <1>      lodsb
1668 0000795C 3C20      <1>      cmp      al, 20h
1669 0000795E 74FB      <1>      je       short get_prompt_name_fchar
1670 00007960 7712      <1>      ja       short loc_change_prompt_label
1671 00007962 BE[FBFA0000] <1>      mov      esi, TRDOSPromptLabel
1672 00007967 C7065452444F <1>      mov      dword [esi], "TRDO"
1673 0000796D 66C746045300 <1>      mov      word [esi+4], "S"
1674      <1>      loc_cmd_prompt_return:
1675 00007973 C3      <1>      retn
1676      <1>      loc_change_prompt_label:
1677 00007974 66B90B00 <1>      mov      cx, 11
1678 00007978 BF[FBFA0000] <1>      mov      edi, TRDOSPromptLabel
1679      <1>      put_char_new_prompt_label:
1680 0000797D AA      <1>      stosb
1681 0000797E AC      <1>      lodsb
1682 0000797F 3C20      <1>      cmp      al, 20h
1683 00007981 7202      <1>      jnb      short pass_put_new_prompt_label
1684 00007983 E2F8      <1>      loop     put_char_new_prompt_label
1685      <1>      pass_put_new_prompt_label:
1686 00007985 C60700 <1>      mov      byte [edi], 0
1687 00007988 C3      <1>      retn
1688      <1>
1689      <1>      cmp_cmd_volume:

```



```

1690 00007989 B106      <1>      mov     cl, 6
1691 0000798B BF[21FB0000] <1>      mov     edi, Cmd_Volume
1692 00007990 E8CB000000 <1>      call    cmp_cmd
1693 00007995 7255      <1>      jc      short cmp_cmd_attrib
1694                                <1>
1695                                <1> cmd_vol1:
1696 00007997 AC          <1>      lodsb
1697 00007998 3C20      <1>      cmp     al, 20h
1698 0000799A 7707      <1>      ja      short cmd_vol2
1699 0000799C A0[E6380100] <1>      mov     al, [Current_Drv]
1700 000079A1 EB3D      <1>      jmp     short cmd_vol4
1701                                <1> cmd_vol2:
1702 000079A3 3C41      <1>      cmp     al, 'A'
1703 000079A5 0F82A2000000 <1>      jnb     loc_cmd_failed
1704 000079AB 3C7A      <1>      cmp     al, 'z'
1705 000079AD 0F879A000000 <1>      ja      loc_cmd_failed
1706 000079B3 3C5A      <1>      cmp     al, 'Z'
1707 000079B5 760A      <1>      jna     short cmd_vol3
1708 000079B7 3C61      <1>      cmp     al, 'a'
1709 000079B9 0F828E000000 <1>      jnb     loc_cmd_failed
1710 000079BF 24DF      <1>      and     al, 0DFh
1711                                <1> cmd_vol3:
1712 000079C1 8A26      <1>      mov     ah, [esi]
1713 000079C3 80FC3A      <1>      cmp     ah, ':'
1714 000079C6 0F8581000000 <1>      jne     loc_cmd_failed
1715 000079CC 2C41      <1>      sub     al, 'A'
1716 000079CE 3A05[A7FA0000] <1>      cmp     al, [Last_DOS_DiskNo]
1717 000079D4 760A      <1>      jna     short cmd_vol4
1718                                <1>
1719 000079D6 BE[67FD0000] <1>      mov     esi, Msg_Not_Ready_Read_Err
1720 000079DB E9DDE8FFFF <1>      jmp     print_msg
1721                                <1>
1722                                <1> cmd_vol4:
1723 000079E0 E896FAFFFF <1>      call    print_volume_info
1724 000079E5 0F8278FEFFFF <1>      jc      cd_drive_not_ready
1725 000079EB C3          <1>      retn
1726                                <1>
1727                                <1> cmp_cmd_attrib:
1728 000079EC B106      <1>      mov     cl, 6
1729 000079EE BF[50FB0000] <1>      mov     edi, Cmd_Attrib
1730 000079F3 E868000000 <1>      call    cmp_cmd
1731 000079F8 0F83310F0000 <1>      jnc     set_file_attributes
1732                                <1>
1733                                <1> cmp_cmd_rename:
1734 000079FE B106      <1>      mov     cl, 6
1735 00007A00 BF[57FB0000] <1>      mov     edi, Cmd_Rename
1736 00007A05 E856000000 <1>      call    cmp_cmd
1737 00007A0A 0F8367110000 <1>      jnc     rename_file
1738                                <1>
1739                                <1> cmp_cmd_device:
1740 00007A10 B106      <1>      mov     cl, 6
1741 00007A12 BF[8CFB0000] <1>      mov     edi, Cmd_Device
1742 00007A17 E844000000 <1>      call    cmp_cmd
1743 00007A1C 7225      <1>      jc      short cmp_cmd_external
1744                                <1>
1745 00007A1E C3          <1>      retn
1746                                <1>
1747                                <1> c_7:
1748                                <1> cmp_cmd_devlist:
1749 00007A1F BF[93FB0000] <1>      mov     edi, Cmd_DevList
1750 00007A24 E837000000 <1>      call    cmp_cmd
1751 00007A29 7218      <1>      jc      short cmp_cmd_external
1752                                <1>
1753                                <1> loc_cmd_return:
1754 00007A2B C3          <1>      retn
1755                                <1>
1756                                <1> c_8:
1757 00007A2C 80F908      <1>      cmp     cl, 8
1758 00007A2F 7712      <1>      ja      short cmp_cmd_external
1759 00007A31 72EC      <1>      jnb     short c_7
1760                                <1>
1761                                <1> cmp_cmd_longname:
1762 00007A33 BF[28FB0000] <1>      mov     edi, Cmd_LongName
1763 00007A38 E823000000 <1>      call    cmp_cmd
1764 00007A3D 0F8342060000 <1>      jnc     get_and_print_longname
1765                                <1>
1766                                <1> cmp_cmd_external:
1767                                <1>      ; 07/05/2016
1768                                <1>      ; 22/04/2016
1769 00007A43 BE[96390100] <1>      mov     esi, CommandBuffer
1770 00007A48 E9051D0000 <1>      jmp     loc_run_check_filename
1771                                <1>
1772                                <1> loc_cmd_failed:
1773 00007A4D 803D[96390100]20 <1>      cmp     byte [CommandBuffer], 20h
1774 00007A54 76D5      <1>      jna     short loc_cmd_return
1775 00007A56 BE[48FD0000] <1>      mov     esi, Msg_Bad_Command
1776                                <1> ; call print_msg
1777                                <1> ;loc_cmd_return:
1778                                <1> ; retn
1779 00007A5B E95DE8FFFF <1>      jmp     print_msg
1780                                <1>
1781                                <1> cmp_cmd:
1782                                <1>      ; 29/01/2016 (TRDOS 386 = TRDOS v2.0)
1783 00007A60 BE[96390100] <1>      mov     esi, CommandBuffer
1784                                <1>      ; edi = internal command word (ASCIIIZ)
1785                                <1>      ; ecx = command length (<=8)
1786                                <1> cmp_cmd_1:
1787 00007A65 AC          <1>      lodsb
1788 00007A66 AE          <1>      scasb
1789 00007A67 750D      <1>      jne     short cmp_cmd_3
1790 00007A69 E2FA      <1>      loop    cmp_cmd_1
1791 00007A6B AC          <1>      lodsb

```

```

1792 00007A6C 3C20      <1>      cmp     al, 20h
1793 00007A6E 7703      <1>      ja      short cmp_cmd_2
1794 00007A70 30C0      <1>      xor     al, al
1795                      <1>      ; ZF = 1 -> internal command word matches
1796 00007A72 C3        <1>      retn
1797                      <1>  cmp_cmd_2:
1798                      <1>      ; ZF = 0 (CF = 0) -> external command word
1799 00007A73 58        <1>      pop     eax ; no return to the caller from here
1800 00007A74 EBCD      <1>      jmp     cmp_cmd_external
1801                      <1>  cmp_cmd_3:
1802 00007A76 F9        <1>      stc
1803                      <1>      ; CF = 1 -> internal command word does not match
1804 00007A77 C3        <1>      retn
1805                      <1>
1806                      <1>  loc_run_cmd_failed:
1807                      <1>      ; 15/03/2016
1808                      <1>      ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
1809                      <1>      ; 07/12/2009 (CMD_INTR.ASM)
1810                      <1>      ; 29/11/2009
1811                      <1>
1812 00007A78 E855000000 <1>      call    restore_cdir_after_cmd_fail
1813                      <1>
1814                      <1>  loc_run_cmd_failed_cmp_al:
1815                      <1>      ; End of Restore_CDIRE code (29/11/2009)
1816                      <1>
1817 00007A7D 3C01      <1>      cmp     al, 1 ; Bad command or file name
1818 00007A7F 74CC      <1>      je      loc_cmd_failed
1819                      <1>  loc_run_dir_not_found:
1820 00007A81 3C03      <1>      cmp     al, 3
1821 00007A83 750A      <1>      jne     short loc_run_file_notfound_msg
1822                      <1>      ; Path not found (MS-DOS Error Code = 3)
1823 00007A85 BE[AAFD0000] <1>      mov     esi, Msg_Dir_Not_Found
1824 00007A8A E92EE8FFFF <1>      jmp     print_msg
1825                      <1>
1826                      <1>  loc_run_file_notfound_msg:
1827 00007A8F 3C02      <1>      cmp     al, 2 ; File not found
1828 00007A91 750A      <1>      jne     short loc_run_file_drv_read_err
1829                      <1>
1830                      <1>  loc_print_file_notfound_msg:
1831 00007A93 BE[C1FD0000] <1>      mov     esi, Msg_File_Not_Found
1832                      <1>      ;call proc_printmsg
1833                      <1>      ;retn
1834 00007A98 E920E8FFFF <1>      jmp     print_msg
1835                      <1>
1836                      <1>  loc_run_file_drv_read_err:
1837                      <1>      ; Err: 17 (Read fault)
1838 00007A9D 3C11      <1>      cmp     al, 17 ; Drive not ready or read error
1839 00007A9F 7404      <1>      je      short loc_run_file_print_drv_read_err
1840                      <1>      ;
1841 00007AA1 3C0F      <1>      cmp     al, 15 ; Drive not ready (or read error)
1842 00007AA3 750A      <1>      jne     short loc_run_file_toobig
1843                      <1>
1844                      <1>  loc_run_file_print_drv_read_err:
1845 00007AA5 BE[67FD0000] <1>      mov     esi, Msg_Not_Ready_Read_Err
1846 00007AAA E90EE8FFFF <1>      jmp     print_msg
1847                      <1>
1848                      <1>  loc_run_file_toobig:
1849 00007AAF 3C08      <1>      cmp     al, 8 ; Not enough free memory to load&run file
1850 00007AB1 750A      <1>      jne     short loc_run_misc_error
1851 00007AB3 BE[0CFE0000] <1>      mov     esi, Msg_Insufficient_Memory
1852 00007AB8 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 00007ABD E807B8FFFF <1>      call    bytetohe
1859 00007AC2 66A3[40FE0000] <1>      mov     [error_code_hex], ax
1860                      <1>
1861 00007AC8 BE[23FE0000] <1>      mov     esi, Msg_Error_Code
1862                      <1>      ;call print_msg
1863                      <1>      ;retn
1864                      <1>
1865 00007ACD 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 00007AD2 50        <1>      push    eax
1870 00007AD3 8A3D[46410100] <1>      mov     bh, [RUN_CDRV] ; it is set at the beginning
1871                      <1>      ; of the 'run' command.
1872 00007AD9 3A3D[E6380100] <1>      cmp     bh, [Current_Drv]
1873 00007ADF 7409      <1>      je      short loc_run_restore_cdir
1874 00007AE1 88FA      <1>      mov     dl, bh
1875 00007AE3 E8D5F0FFFF <1>      call    change_current_drive
1876 00007AE8 EB19      <1>      jmp     short loc_run_err_pass_restore_cdir
1877                      <1>
1878                      <1>  loc_run_restore_cdir:
1879 00007AEA 803D[A8FA0000]00 <1>      cmp     byte [Restore_CDIRE], 0
1880 00007AF1 7610      <1>      jna     short loc_run_err_pass_restore_cdir
1881 00007AF3 30DB      <1>      xor     bl, bl
1882 00007AF5 0FB7F3      <1>      movzx    esi, bx
1883 00007AF8 81C600010900 <1>      add     esi, Logical_DOSDisks
1884 00007AFE E871F1FFFF <1>      call    restore_current_directory
1885                      <1>
1886                      <1>  loc_run_err_pass_restore_cdir:
1887 00007B03 58        <1>      pop     eax
1888 00007B04 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 00007B05 66C705[88420100]00- <1>      mov     word [AttributesMask], 0800h ; ..except volume names..
1895 00007B0D 08                    <1>
1896 00007B0E A0[E6380100]         <1>      mov     al, [Current_Drv]
1897 00007B13 A2[46410100]         <1>      mov     [RUN_CDRV], al
1898                                     <1> get_dfname_fchar:
1899 00007B18 AC                    <1>      lodsb
1900 00007B19 3C20                  <1>      cmp     al, 20h
1901 00007B1B 74FB                  <1>      je      short get_dfname_fchar
1902 00007B1D 0F82A4000000          <1>      jnb     loc_print_dir_call_all
1903 00007B23 3C2D                  <1>      cmp     al, '-'
1904 00007B25 7542                  <1>      jne     short loc_print_dir_call_flt
1905                                     <1> get_next_attr_char:
1906 00007B27 AC                    <1>      lodsb
1907 00007B28 3C20                  <1>      cmp     al, 20h
1908 00007B2A 74FB                  <1>      je      short get_next_attr_char
1909 00007B2C 0F821BFFFFFF          <1>      jnb     loc_cmd_failed
1910 00007B32 24DF                  <1>      and     al, 0DFh
1911 00007B34 3C44                  <1>      cmp     al, 'D' ; directories only ?
1912 00007B36 7512                  <1>      jne     short pass_only_directories
1913 00007B38 AC                    <1>      lodsb
1914 00007B39 3C20                  <1>      cmp     al, 20h
1915 00007B3B 0F870CFFFFFF          <1>      ja      loc_cmd_failed
1916 00007B41 800D[88420100]10      <1>      or      byte [AttributesMask], 10h ; ..directory..
1917 00007B48 EB18                  <1>      jmp     short get_dfname_fchar_attr
1918                                     <1> pass_only_directories:
1919 00007B4A 3C46                  <1>      cmp     al, 'F' ; files only ?
1920 00007B4C 0F85B0000000          <1>      jne     check_attr_s
1921 00007B52 AC                    <1>      lodsb
1922 00007B53 3C20                  <1>      cmp     al, 20h
1923 00007B55 0F87F2FEFFFFFF          <1>      ja      loc_cmd_failed
1924 00007B5B 800D[89420100]10      <1>      or      byte [AttributesMask+1], 10h ; ..except directories..
1925                                     <1> get_dfname_fchar_attr:
1926 00007B62 AC                    <1>      lodsb
1927 00007B63 3C20                  <1>      cmp     al, 20h
1928 00007B65 74FB                  <1>      je      short get_dfname_fchar_attr
1929 00007B67 725E                  <1>      jnb     short loc_print_dir_call_all
1930                                     <1>
1931                                     <1> loc_print_dir_call_flt:
1932 00007B69 4E                    <1>      dec     esi
1933 00007B6A BF[8A420100]          <1>      mov     edi, FindFile_Drv
1934 00007B6F E8F2250000            <1>      call    parse_path_name
1935 00007B74 7308                  <1>      jnc     short loc_print_dir_change_drv_1
1936 00007B76 3C01                  <1>      cmp     al, 1
1937 00007B78 0F87FAFEFFFFFF          <1>      ja      loc_run_cmd_failed
1938                                     <1>
1939                                     <1> loc_print_dir_change_drv_1:
1940 00007B7E 8A15[8A420100]          <1>      mov     dl, [FindFile_Drv]
1941                                     <1> loc_print_dir_change_drv_2:
1942 00007B84 3A15[46410100]          <1>      cmp     dl, [RUN_CDRV]
1943 00007B8A 740B                  <1>      je      short loc_print_dir_change_directory
1944 00007B8C E82CF0FFFF            <1>      call    change_current_drive
1945 00007B91 0F82E1FEFFFFFF          <1>      jc      loc_run_cmd_failed
1946                                     <1> loc_print_dir_change_directory:
1947 00007B97 803D[8B420100]20        <1>      cmp     byte [FindFile_Directory], 20h ; 0 or 20h ?
1948 00007B9E 761D                  <1>      jna     short pass_print_dir_change_directory
1949                                     <1>
1950 00007BA0 FE05[A8FA0000]          <1>      inc     byte [Restore_CDIR]
1951 00007BA6 BE[8B420100]          <1>      mov     esi, FindFile_Directory
1952 00007BAB 30E4                  <1>      xor     ah, ah ; CD_COMMAND sign -> 0
1953 00007BAD E8A01F0000            <1>      call    change_current_directory
1954 00007BB2 0F82C0FEFFFFFF          <1>      jc      loc_run_cmd_failed
1955                                     <1>
1956                                     <1> loc_print_dir_change_prompt_dir_string:
1957 00007BB8 E8B51E0000            <1>      call    change_prompt_dir_string
1958                                     <1>
1959                                     <1> pass_print_dir_change_directory:
1960 00007BBD BE[CC420100]          <1>      mov     esi, FindFile_Name
1961 00007BC2 803E20                  <1>      cmp     byte [esi], 20h ; ; 0 or 20h ?
1962 00007BC5 7706                  <1>      ja      short loc_print_dir_call
1963                                     <1>
1964                                     <1> loc_print_dir_call_all:
1965 00007BC7 C7062A2E2A00            <1>      mov     dword [esi], '*.*'
1966                                     <1> loc_print_dir_call:
1967 00007BCD E87E000000            <1>      call    print_directory
1968                                     <1>
1969 00007BD2 8A15[46410100]          <1>      mov     dl, [RUN_CDRV] ; it is set at the beginning
1970 00007BD8 3A15[E6380100]          <1>      cmp     dl, [Current_Drv]
1971 00007BDE 7406                  <1>      je      short loc_print_dir_call_restore_cdir_retn
1972 00007BE0 E8D8EFFFFFFF            <1>      call    change_current_drive
1973 00007BE5 C3                    <1>      retn
1974                                     <1>
1975                                     <1> loc_print_dir_call_restore_cdir_retn:
1976 00007BE6 803D[A8FA0000]00        <1>      cmp     byte [Restore_CDIR], 0
1977 00007BED 7610                  <1>      jna     short pass_print_dir_call_restore_cdir_retn
1978                                     <1>
1979 00007BEF BE00010900            <1>      mov     esi, Logical_DOSDisks
1980 00007BF4 31C0                  <1>      xor     eax, eax
1981 00007BF6 88D4                  <1>      mov     ah, dl
1982 00007BF8 01C6                  <1>      add     esi, eax
1983                                     <1>
1984 00007BFA E875F0FFFF            <1>      call    restore_current_directory
1985                                     <1>
1986                                     <1> pass_print_dir_call_restore_cdir_retn:
1987 00007BFF C3                    <1>      retn
1988                                     <1>
1989                                     <1> check_attr_s_cap:
1990 00007C00 24DF                  <1>      and     al, 0DFh
1991                                     <1> check_attr_s:
1992 00007C02 3C53                  <1>      cmp     al, 'S'
1993 00007C04 7514                  <1>      jne     short pass_attr_s
1994 00007C06 800D[88420100]04        <1>      or      byte [AttributesMask], 4 ; system

```

```

1995 00007C0D AC          <1>      lodsb
1996 00007C0E 3C20        <1>      cmp     al, 20h
1997 00007C10 0F844CFFFFFF <1>      je      get_dfname_fchar_attr
1998 00007C16 72AF        <1>      jnb     short loc_print_dir_call_all
1999 00007C18 24DF        <1>      and     al, 0DFh
2000                                <1> pass_attr_s:
2001 00007C1A 3C48        <1>      cmp     al, 'H'
2002 00007C1C 7514        <1>      jne     short pass_attr_h
2003 00007C1E 800D[88420100]02 <1>      or      byte [AttributesMask], 2 ; hidden
2004                                <1> pass_attr_shr:
2005 00007C25 AC          <1>      lodsb
2006 00007C26 3C20        <1>      cmp     al, 20h
2007 00007C28 0F8434FFFFFF <1>      je      get_dfname_fchar_attr
2008 00007C2E 7297        <1>      jnb     short loc_print_dir_call_all
2009 00007C30 EBCE        <1>      jmp     short check_attr_s_cap
2010                                <1>
2011                                <1> pass_attr_h:
2012 00007C32 3C52        <1>      cmp     al, 'R'
2013 00007C34 7509        <1>      jne     short pass_attr_r
2014 00007C36 800D[88420100]01 <1>      or      byte [AttributesMask], 1 ; read only
2015 00007C3D EBE6        <1>      jmp     short pass_attr_shr
2016                                <1>
2017                                <1> pass_attr_r:
2018 00007C3F 3C41        <1>      cmp     al, 'A'
2019 00007C41 0F8506FFFFFF <1>      jne     loc_cmd_failed
2020 00007C47 800D[88420100]20 <1>      or      byte [AttributesMask], 20h ; archive
2021 00007C4E 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 00007C50 56          <1>      push    esi
2035                                <1>
2036 00007C51 29C0        <1>      sub     eax, eax
2037                                <1>
2038 00007C53 66A3[14430100] <1>      mov     word [Dir_Count], ax ; 0
2039 00007C59 66A3[12430100] <1>      mov     word [File_Count], ax ; 0
2040 00007C5F A3[16430100] <1>      mov     dword [Total_FSize], eax ; 0
2041                                <1>
2042 00007C64 E86AE6FFFF <1>      call    clear_screen
2043                                <1>
2044 00007C69 31C9        <1>      xor     ecx, ecx
2045 00007C6B 8A2D[E6380100] <1>      mov     ch, [Current_Drv] ; DirBuff_Drv - 'A'
2046 00007C71 A0[E7380100] <1>      mov     al, [Current_Dir_Drv]
2047 00007C76 A2[65FC0000] <1>      mov     [Dir_Drive_Name], al
2048 00007C7B BE00010900 <1>      mov     esi, Logical_DOSDisks
2049 00007C80 01CE        <1>      add     esi, ecx
2050                                <1>
2051 00007C82 E86EF9FFFF <1>      call    move_volume_name_and_serial_no
2052 00007C87 730C        <1>      jnc     short print_dir_strlen_check
2053                                <1>
2054 00007C89 5E          <1>      pop     esi
2055 00007C8A 8A3D[4E380100] <1>      mov     bh, [ptty] ; [ACTIVE_PAGE]
2056                                <1>      ;call beeper
2057                                <1>      ;retn
2058 00007C90 E9FDA0FFFF <1>      jmp     beeper ; beep ! and return
2059                                <1>
2060                                <1> print_dir_strlen_check:
2061 00007C95 BE[E9380100] <1>      mov     esi, Current_Dir_Root
2062 00007C9A BF[02FD0000] <1>      mov     edi, Dir_Str_Root
2063                                <1>
2064                                <1>      ;xor ecx, ecx
2065 00007C9F 8A0D[45390100] <1>      mov     cl, [Current_Dir_StrLen]
2066 00007CA5 FEC1        <1>      inc     cl
2067 00007CA7 80F940        <1>      cmp     cl, 64
2068 00007CAA 760D        <1>      jna     short pass_print_dir_strlen_shorting
2069 00007CAC 46          <1>      inc     esi
2070 00007CAD 01CE        <1>      add     esi, ecx
2071 00007CAF 83EE40        <1>      sub     esi, 64
2072 00007CB2 47          <1>      inc     edi
2073 00007CB3 B82E2E2E20 <1>      mov     eax, '... '
2074 00007CB8 AB          <1>      stosd
2075                                <1>
2076                                <1> pass_print_dir_strlen_shorting:
2077 00007CB9 F3A4        <1>      rep     movsb
2078                                <1>
2079 00007CBB BE[58FC0000] <1>      mov     esi, Dir_Drive_Str
2080 00007CC0 E8F8E5FFFF <1>      call    print_msg
2081                                <1>
2082 00007CC5 BE[B7FC0000] <1>      mov     esi, Vol_Serial_Header
2083 00007CCA E8EEE5FFFF <1>      call    print_msg
2084                                <1>
2085 00007CCF BE[F7FC0000] <1>      mov     esi, Dir_Str_Header
2086 00007CD4 E8E4E5FFFF <1>      call    print_msg
2087                                <1>
2088 00007CD9 BE[11070100] <1>      mov     esi, next2line
2089 00007CDE E8DAE5FFFF <1>      call    print_msg
2090                                <1>
2091                                <1> loc_print_dir_first_file:
2092 00007CE3 C605[29430100]10 <1>      mov     byte [PrintDir_RowCounter], 16
2093 00007CEA 66A1[88420100] <1>      mov     ax, [AttributesMask]
2094 00007CF0 5E          <1>      pop     esi
2095                                <1>
2096 00007CF1 E859020000 <1>      call    find_first_file

```



```

2097 00007CF6 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 00007CFC F6C310          <1>         test    bl, 10h ; Is it a directory?
2102 00007CFF 741B            <1>         jz      short loc_not_dir
2103                                <1>
2104 00007D01 66FF05[14430100] <1>         inc     word [Dir_Count]
2105 00007D08 89F2            <1>         mov     edx, esi ; FindFile_DirEntry address
2106 00007D0A BE[46FE0000]    <1>         mov     esi, Type_Dir; '<DIR>'
2107 00007D0F BF[5DFE0000]    <1>         mov     edi, Dir_Or_FileSize
2108                                <1>         ; move 10 bytes
2109 00007D14 A5              <1>         movsd
2110 00007D15 A5              <1>         movsd
2111 00007D16 66A5            <1>         movsw
2112 00007D18 89D6            <1>         mov     esi, edx
2113 00007D1A EB36            <1>         jmp     short loc_dir_attribute
2114                                <1>
2115                                <1> loc_not_dir:
2116 00007D1C 66FF05[12430100] <1>         inc     word [File_Count]
2117 00007D23 0105[16430100] <1>         add     [Total_FSize], eax
2118                                <1>
2119 00007D29 B90A000000      <1>         mov     ecx, 10 ; 32 bit divisor
2120 00007D2E 89CF            <1>         mov     edi, ecx
2121 00007D30 81C7[5DFE0000] <1>         add     edi, Dir_Or_FileSize
2122                                <1> loc_dir_rdivide:
2123 00007D36 29D2            <1>         sub     edx, edx
2124 00007D38 F7F1            <1>         div     ecx ; remainder in dl (< 10)
2125 00007D3A 80C230          <1>         add     dl, '0' ; to make visible (ascii)
2126 00007D3D 4F              <1>         dec     edi
2127 00007D3E 8817            <1>         mov     [edi], dl
2128 00007D40 21C0            <1>         and     eax, eax
2129 00007D42 75F2            <1>         jnz     short loc_dir_rdivide
2130                                <1>
2131                                <1> loc_dir_fill_space:
2132 00007D44 81FF[5DFE0000] <1>         cmp     edi, Dir_Or_FileSize
2133 00007D4A 7606            <1>         jna     short loc_dir_attribute
2134 00007D4C 4F              <1>         dec     edi
2135 00007D4D C60720          <1>         mov     byte [edi], 20h
2136 00007D50 EBF2            <1>         jmp     short loc_dir_fill_space
2137                                <1>
2138                                <1> loc_dir_attribute:
2139 00007D52 C705[68FE0000]2020- <1>         mov     dword [File_Attribute], 20202020h
2139 00007D5A 2020            <1>
2140                                <1>
2141 00007D5C 80FB20          <1>         cmp     bl, 20h ; Is it an archive file?
2142 00007D5F 7207            <1>         jb     short loc_dir_pass_arch
2143 00007D61 C605[6BFE0000]41 <1>         mov     byte [File_Attribute+3], 'A'
2144                                <1>
2145                                <1> loc_dir_pass_arch:
2146 00007D68 80E307          <1>         and     bl, 7
2147 00007D6B 7428            <1>         jz     short loc_dir_file_name
2148 00007D6D 88DF            <1>         mov     bh, bl
2149 00007D6F 80E303          <1>         and     bl, 3
2150 00007D72 38DF            <1>         cmp     bh, bl
2151 00007D74 7607            <1>         jna     short loc_dir_pass_s
2152 00007D76 C605[68FE0000]53 <1>         mov     byte [File_Attribute], 'S'
2153                                <1>
2154                                <1> loc_dir_pass_s:
2155 00007D7D 80E302          <1>         and     bl, 2
2156 00007D80 7407            <1>         jz     short loc_dir_pass_h
2157 00007D82 C605[69FE0000]48 <1>         mov     byte [File_Attribute+1], 'H'
2158                                <1> loc_dir_pass_h:
2159 00007D89 80E701          <1>         and     bh, 1
2160 00007D8C 7407            <1>         jz     short loc_dir_file_name
2161 00007D8E C605[6AFE0000]52 <1>         mov     byte [File_Attribute+2], 'R'
2162                                <1> loc_dir_file_name:
2163                                <1>         ;mov     bx, [esi+18h] ; Date
2164                                <1>         ;mov     dx, [esi+16h] ; Time
2165 00007D95 8B5E16          <1>         mov     ebx, [esi+16h]
2166 00007D98 89F1            <1>         mov     ecx, esi ; FindFile_DirEntry address
2167 00007D9A BF[50FE0000]    <1>         mov     edi, File_Name
2168                                <1>         ; move 8 bytes
2169 00007D9F A5              <1>         movsd
2170 00007DA0 A5              <1>         movsd
2171 00007DA1 C60720          <1>         mov     byte [edi], 20h
2172 00007DA4 47              <1>         inc     edi
2173                                <1>         ; move 3 bytes
2174 00007DA5 66A5            <1>         movsw
2175 00007DA7 A4              <1>         movsb
2176 00007DA8 89CE            <1>         mov     esi, ecx
2177                                <1>
2178                                <1> Dir_Time_start:
2179                                <1>         ;mov     ax, dx ; Time
2180 00007DAA 6689D8          <1>         mov     ax, bx
2181 00007DAD 66C1E805        <1>         shr     ax, 5 ; shift right 5 times
2182 00007DB1 6683E03F        <1>         and     ax, 000011111b ; Minute Mask
2183 00007DB5 D40A            <1>         aam ; Q([AL]/10)->AH
2184                                <1>         ; R([AL]/10)->AL
2185                                <1>         ; [AL]+[AH]= Minute as BCD
2186 00007DB7 660D3030        <1>         or      ax, '00' ; Convert to ASCII
2187 00007DBB 86E0            <1>         xchg    ah, al
2188 00007DBD 66A3[7BFE0000] <1>         mov     [File_Minute], ax
2189                                <1>
2190                                <1>         ;mov     al, dh
2191 00007DC3 88F8            <1>         mov     al, bh
2192 00007DC5 C0E803          <1>         shr     al, 3 ; shift right 3 times
2193 00007DC8 D40A            <1>         aam ; [AL]+[AH]= Hours as BCD
2194 00007DCA 660D3030        <1>         or      ax, '00'
2195 00007DCE 86E0            <1>         xchg    ah, al
2196 00007DD0 66A3[78FE0000] <1>         mov     [File_Hour], ax
2197                                <1>

```

```

2198 00007DD6 C1EB10      <1>      shr     ebx, 16          ; BX = Date
2199                      <1>
2200                      <1> Dir_Date_start:
2201 00007DD9 6689D8      <1>      mov     ax, bx          ; Date
2202 00007DDC 6683E01F    <1>      and     ax, 00011111b; Day Mask
2203 00007DE0 D40A       <1>      aam                     ; Q([AL]/10)->AH
2204                      <1>                      ; R([AL]/10)->AL
2205                      <1>                      ; [AL]+[AH]= Day as BCD
2206 00007DE2 660D3030    <1>      or      ax, '00'       ; Convert to ASCII
2207 00007DE6 86C4       <1>      xchg    al, ah
2208                      <1>
2209 00007DE8 66A3[6DFE0000] <1>      mov     [File_Day], ax
2210                      <1>
2211 00007DEE 6689D8      <1>      mov     ax, bx
2212 00007DF1 66C1E805    <1>      shr     ax, 5          ; shift right 5 times
2213 00007DF5 6683E00F    <1>      and     ax, 00001111b; Month Mask
2214 00007DF9 D40A       <1>      aam
2215 00007DFB 660D3030    <1>      or      ax, '00'
2216 00007DFF 86E0       <1>      xchg    ah, al
2217 00007E01 66A3[70FE0000] <1>      mov     [File_Month], ax
2218                      <1>
2219 00007E07 6689D8      <1>      mov     ax, bx
2220 00007E0A 66C1E809    <1>      shr     ax, 9
2221 00007E0E 6683E07F    <1>      and     ax, 01111111b; Result = Year - 1980
2222 00007E12 6605BC07    <1>      add     ax, 1980
2223                      <1>
2224 00007E16 B10A       <1>      mov     cl, 10
2225 00007E18 F6F1       <1>      div     cl          ; Q -> AL, R -> AH
2226 00007E1A 80CC30      <1>      or      ah, '0'
2227 00007E1D 8825[76FE0000] <1>      mov     [File_Year+3], ah
2228 00007E23 D40A       <1>      aam
2229 00007E25 86E0       <1>      xchg    ah, al
2230 00007E27 80CC30      <1>      or      ah, '0'       ; Convert to ASCII
2231 00007E2A 8825[75FE0000] <1>      mov     [File_Year+2], ah
2232 00007E30 D40A       <1>      aam
2233 00007E32 86C4       <1>      xchg    al, ah
2234 00007E34 660D3030    <1>      or      ax, '00'
2235 00007E38 66A3[73FE0000] <1>      mov     [File_Year], ax
2236                      <1>
2237                      <1> loc_show_line:
2238 00007E3E 56          <1>      push    esi
2239 00007E3F BE[50FE0000] <1>      mov     esi, File_Name
2240 00007E44 E874E4FFFF    <1>      call    print_msg
2241 00007E49 BE[13070100] <1>      mov     esi, nextline
2242 00007E4E E86AE4FFFF    <1>      call    print_msg
2243 00007E53 5E          <1>      pop     esi
2244                      <1>
2245 00007E54 FE0D[29430100] <1>      dec     byte [PrintDir_RowCounter]
2246 00007E5A 0F84D4000000 <1>      jz      pause_dir_scroll
2247                      <1>
2248                      <1> loc_next_entry:
2249 00007E60 E899010000    <1>      call    find_next_file
2250 00007E65 0F8391FEFFFF    <1>      jnc     loc_dfname_use_this
2251                      <1>
2252                      <1> loc_dir_ok:
2253 00007E6B B90A000000    <1>      mov     ecx, 10
2254 00007E70 66A1[14430100] <1>      mov     ax, [Dir_Count]
2255 00007E76 BF[91FE0000]   <1>      mov     edi, Decimal_Dir_Count
2256 00007E7B 6639C8      <1>      cmp     ax, cx ; 10
2257 00007E7E 7216       <1>      jb      short pass_ddc
2258 00007E80 47          <1>      inc     edi
2259 00007E81 6683F864    <1>      cmp     ax, 100
2260 00007E85 720F       <1>      jb      short pass_ddc
2261 00007E87 47          <1>      inc     edi
2262 00007E88 663DE803    <1>      cmp     ax, 1000
2263 00007E8C 7208       <1>      jb      short pass_ddc
2264 00007E8E 47          <1>      inc     edi
2265 00007E8F 663D1027    <1>      cmp     ax, 10000
2266 00007E93 7201       <1>      jb      short pass_ddc
2267 00007E95 47          <1>      inc     edi
2268                      <1> pass_ddc:
2269 00007E96 886F01      <1>      mov     [edi+1], ch ; 0
2270                      <1> loc_ddc_rediv:
2271 00007E99 31D2       <1>      xor     edx, edx
2272 00007E9B 66F7F1      <1>      div     cx ; 10
2273 00007E9E 80C230      <1>      add     dl, '0'
2274 00007EA1 8817       <1>      mov     [edi], dl
2275 00007EA3 4F          <1>      dec     edi
2276 00007EA4 6609C0      <1>      or      ax, ax
2277 00007EA7 75F0       <1>      jnz     short loc_ddc_rediv
2278                      <1>
2279 00007EA9 66A1[12430100] <1>      mov     ax, [File_Count]
2280 00007EAF BF[80FE0000]   <1>      mov     edi, Decimal_File_Count
2281 00007EB4 6639C8      <1>      cmp     ax, cx ; 10
2282 00007EB7 7216       <1>      jb      short pass_dfc
2283 00007EB9 47          <1>      inc     edi
2284 00007EBA 6683F864    <1>      cmp     ax, 100
2285 00007EBE 720F       <1>      jb      short pass_dfc
2286 00007EC0 47          <1>      inc     edi
2287 00007EC1 663DE803    <1>      cmp     ax, 1000
2288 00007EC5 7208       <1>      jb      short pass_dfc
2289 00007EC7 47          <1>      inc     edi
2290 00007EC8 663D1027    <1>      cmp     ax, 10000
2291 00007ECC 7201       <1>      jb      short pass_dfc
2292 00007ECE 47          <1>      inc     edi
2293                      <1> pass_dfc:
2294                      <1> ;mov     cx, 10
2295 00007ECF 886F01      <1>      mov     [edi+1], ch ; 00
2296                      <1> loc_dfc_rediv:
2297                      <1> ;xor     dx, dx
2298 00007ED2 30D2       <1>      xor     dl, dl
2299 00007ED4 66F7F1      <1>      div     cx

```

```

2300 00007ED7 80C230      <1>      add     dl, '0'
2301 00007EDA 8817      <1>      mov     [edi], dl
2302 00007EDC 4F        <1>      dec     edi
2303 00007EDD 6609C0    <1>      or      ax, ax
2304 00007EE0 75F0      <1>      jnz     short loc_dfc_rediv
2305                                <1>
2306 00007EE2 BF[28430100] <1>      mov     edi, TFS_Dec_End
2307                                <1>      ;mov     byte [edi], 0
2308 00007EE7 A1[16430100] <1>      mov     eax, [Total_FSize]
2309                                <1>      ;mov     ecx, 10
2310                                <1> rediv_tfs_hex:
2311                                <1>      ;sub     edx, edx
2312 00007EEC 28D2      <1>      sub     dl, dl
2313 00007EEE F7F1      <1>      div     ecx
2314 00007EF0 80C230    <1>      add     dl, '0'
2315 00007EF3 4F        <1>      dec     edi
2316 00007EF4 8817      <1>      mov     [edi], dl
2317 00007EF6 21C0      <1>      and     eax, eax
2318 00007EF8 75F2      <1>      jnz     short rediv_tfs_hex
2319                                <1>
2320 00007EFA 893D[1A430100] <1>      mov     [TFS_Dec_Begin], edi
2321 00007F00 BE[7EFE0000] <1>      mov     esi, Decimal_File_Count_Header
2322 00007F05 E8B3E3FFFF <1>      call    print_msg
2323 00007F0A BE[86FE0000] <1>      mov     esi, str_files
2324 00007F0F E8A9E3FFFF <1>      call    print_msg
2325 00007F14 BE[97FE0000] <1>      mov     esi, str_dirs
2326 00007F19 E89FE3FFFF <1>      call    print_msg
2327 00007F1E 8B35[1A430100] <1>      mov     esi, [TFS_Dec_Begin]
2328 00007F24 E894E3FFFF <1>      call    print_msg
2329 00007F29 BE[A8FE0000] <1>      mov     esi, str_bytes
2330 00007F2E E88AE3FFFF <1>      call    print_msg
2331                                <1>
2332 00007F33 C3        <1>      retn
2333                                <1>
2334                                <1> pause_dir_scroll:
2335 00007F34 28E4      <1>      sub     ah, ah
2336 00007F36 E8DB8CFFFF <1>      call    int16h
2337 00007F3B 3C1B      <1>      cmp     al, 1Bh
2338 00007F3D 0F8428FFFFFF <1>      je      loc_dir_ok
2339 00007F43 C605[29430100]10 <1>      mov     byte [PrintDir_RowCounter], 16 ; Reset counter
2340 00007F4A 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
2348                                <1>      ; 2005
2349                                <1>      ; INPUT ->
2350                                <1>      ;     ESI = ASCIIZ File/Dir Name Address (in Current Directory)
2351                                <1>      ;     AL = Attributes AND mask (The AND result must be equal to AL)
2352                                <1>      ;           bit 0 = Read Only
2353                                <1>      ;           bit 1 = Hidden
2354                                <1>      ;           bit 2 = System
2355                                <1>      ;           bit 3 = Volume Label
2356                                <1>      ;           bit 4 = Directory
2357                                <1>      ;           bit 5 = Archive
2358                                <1>      ;           bit 6 = Reserved, must be 0
2359                                <1>      ;           bit 7 = Reserved, must be 0
2360                                <1>      ;     AH = Attributes Negative AND mask (The AND result must be ZERO)
2361                                <1>      ;
2362                                <1>      ; OUTPUT ->
2363                                <1>      ;     CF = 1 -> Error, Error Code in EAX (AL)
2364                                <1>      ;     CF = 0 ->
2365                                <1>      ;     ESI = Directory Entry (FindFile_DirEntry) Location
2366                                <1>      ;     EDI = Directory Buffer Directory Entry Location
2367                                <1>      ;     EAX = File Size
2368                                <1>      ;     BL = Attributes of The File/Directory
2369                                <1>      ;     BH = Long Name Yes/No Status (>0 is YES)
2370                                <1>      ;     DX > 0 : Ambiguous filename chars are used
2371                                <1>      ;
2372                                <1>      ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
2373                                <1>
2374 00007F4F 66A3[DA420100] <1>      mov     [FindFile_AttributesMask], ax
2375 00007F55 BF[DC420100] <1>      mov     edi, FindFile_DirEntry ; TR-DOS Fullfilename formatted buffer
2376 00007F5A 31C0      <1>      xor     eax, eax
2377 00007F5C B90B000000 <1>      mov     ecx, 11
2378 00007F61 F3AB      <1>      rep     stosd ; 44 bytes
2379                                <1>      ;stosw     ; +2 bytes
2380                                <1>
2381 00007F63 BF[CC420100] <1>      mov     edi, FindFile_Name ; FFF structure, offset 66
2382 00007F68 39FE      <1>      cmp     esi, edi
2383 00007F6A 7408      <1>      je      short loc_fff_mfn_ok
2384 00007F6C 89FA      <1>      mov     edx, edi
2385                                <1>      ; move 13 bytes
2386 00007F6E A5        <1>      movsd
2387 00007F6F A5        <1>      movsd
2388 00007F70 A5        <1>      movsd
2389 00007F71 AA        <1>      stosb
2390 00007F72 89D6      <1>      mov     esi, edx
2391                                <1> loc_fff_mfn_ok:
2392 00007F74 BF[7B420100] <1>      mov     edi, Dir_Entry_Name ; Dir Entry Format File Name
2393 00007F79 E81D210000 <1>      call    convert_file_name
2394 00007F7E 89FE      <1>      mov     esi, edi ; offset Dir_Entry_Name
2395                                <1>
2396 00007F80 66A1[DA420100] <1>      mov     ax, [FindFile_AttributesMask]
2397                                <1>      ;xor     ecx, ecx
2398 00007F86 30C9      <1>      xor     cl, cl
2399 00007F88 E8191E0000 <1>      call    locate_current_dir_file
2400 00007F8D 726E      <1>      jc      short loc_fff_retn
2401                                <1>      ; EDI = Directory Entry

```

```

2402      <1>      ; EBX = Directory Buffer Entry Index/Number
2403      <1>
2404      <1> loc_fff_fnf_ln_check:
2405      <1>      xor     ch, ch
2406      <1>      xor     dh, 0Fh
2407      <1>      jz      short loc_fff_longname_yes
2408      <1>      mov     [FindFile_LongNameYes], ch ; 0
2409      <1>      jmp     short loc_fff_longname_no
2410      <1>
2411      <1> loc_fff_longname_yes:
2412      <1>      ;inc     byte [FindFile_LongNameYes]
2413      <1>      mov     cl, [LFN_EntryLength]
2414      <1>      mov     [FindFile_LongNameEntryLength], cl ; FindFile_LongNameYes
2415      <1>
2416      <1> loc_fff_longname_no:
2417      <1>      ;mov     bx, [DirBuff_CurrentEntry]
2418      <1>      mov     [FindFile_DirEntryNumber], bx
2419      <1>      mov     dx, ax ; Ambiguous Filename chars used sign > 0
2420      <1>
2421      <1>      mov     al, [Current_Drv]
2422      <1>      mov     [FindFile_Drv], al
2423      <1>
2424      <1>      mov     eax, [Current_Dir_FCluster]
2425      <1>      mov     [FindFile_DirFirstCluster], eax
2426      <1>
2427      <1>      mov     eax, [DirBuff_Cluster]
2428      <1>      mov     [FindFile_DirCluster], eax
2429      <1>
2430      <1>      inc     word [FindFile_MatchCounter]
2431      <1>
2432      <1>      mov     ebx, edi
2433      <1>      mov     esi, edi
2434      <1>      mov     edi, FindFile_DirEntry
2435      <1>      mov     eax, edi
2436      <1>      mov     cl, 8
2437      <1>      rep     movsd
2438      <1>      mov     esi, eax
2439      <1>      mov     edi, ebx
2440      <1>
2441      <1>      mov     eax, [FindFile_DirEntry+28] ; File Size
2442      <1>
2443      <1>      mov     bl, [FindFile_DirEntry+11] ; File Attributes
2444      <1>      mov     bh, [FindFile_LongNameYes]
2445      <1>
2446      <1>      ;mov     cx, [DirBuff_EntryCounter]
2447      <1>      ;mov     [FindFile_DirEntryNumber], cx
2448      <1>      ;mov     cx, [FindFile_DirEntryNumber]
2449      <1>      ; ecx = 0
2450      <1>
2451      <1> loc_fff_retn:
2452      <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      <1>      cmp     word [FindFile_MatchCounter], 0
2476      <1>      ja      short loc_start_search_next_file
2477      <1>
2478      <1> loc_fnf_stc_retn:
2479      <1>      stc
2480      <1> loc_fnf_ax12h_retn:
2481      <1>      mov     eax, 12 ; No More files
2482      <1> ;loc_fnf_retn:
2483      <1>      retn
2484      <1>
2485      <1> loc_start_search_next_file:
2486      <1>      mov     bx, [FindFile_DirEntryNumber]
2487      <1>      inc     bx
2488      <1>      cmp     bx, [DirBuff_LastEntry]
2489      <1>      ja      short loc_cont_search_next_file
2490      <1>
2491      <1> loc_fnf_search:
2492      <1>      mov     esi, Dir_Entry_Name
2493      <1>      mov     ax, [FindFile_AttributesMask]
2494      <1>      xor     cx, cx
2495      <1>      call    find_directory_entry
2496      <1>      jnc     loc_fff_fnf_ln_check
2497      <1>
2498      <1> loc_cont_search_next_file:
2499      <1>      xor     ebx, ebx
2500      <1>      mov     bh, [Current_Drv]
2501      <1>      mov     esi, Logical_DOSDisks
2502      <1>      add     esi, ebx
2503      <1>

```



```

2504 00008049 803D[E4380100]00 <1>      cmp     byte [Current_Dir_Level], 0
2505 00008050 7608 <1>      jna     short loc_fnf_check_FAT_type
2506 00008052 807E0301 <1>      cmp     byte [esi+LD_FATType], 1
2507 00008056 72B1 <1>      jnb     short loc_fnf_ax12h_retn
2508 00008058 EB06 <1>      jmp     short loc_fnf_check_next_cluster
2509 <1>
2510 <1> loc_fnf_check_FAT_type:
2511 0000805A 807E0303 <1>      cmp     byte [esi+LD_FATType], 3
2512 0000805E 72A9 <1>      jnb     short loc_fnf_ax12h_retn
2513 <1>
2514 <1> loc_fnf_check_next_cluster:
2515 00008060 A1[15410100] <1>      mov     eax, [DirBuff_Cluster]
2516 00008065 E813380000 <1>      call    get_next_cluster
2517 0000806A 7306 <1>      jnc     short loc_fnf_load_next_dir_cluster
2518 0000806C 09C0 <1>      or      eax, eax
2519 0000806E 7498 <1>      jz      short loc_fnf_stc_retn
2520 <1>      ;mov     eax, 17 ;Drive not ready or read error
2521 00008070 F5 <1>      cmc     ;stc
2522 <1> loc_fnf_retn:
2523 00008071 C3 <1>      retn
2524 <1>
2525 <1> loc_fnf_load_next_dir_cluster:
2526 00008072 E8EC390000 <1>      call    load_FAT_sub_directory
2527 00008077 72F8 <1>      jc      short loc_fnf_retn
2528 00008079 6631DB <1>      xor     bx, bx
2529 0000807C 66891D[04430100] <1>      mov     [FindFile_DirEntryNumber], bx
2530 00008083 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 00008085 803E20 <1>      cmp     byte [esi], 20h
2539 00008088 7701 <1>      ja      short loc_find_longname
2540 <1>      ;jnb     short loc_longname_retn
2541 <1>      ;inc     esi
2542 <1>      ;je      short get_longname_fchar
2543 <1> ;loc_longname_retn:
2544 0000808A C3 <1>      retn
2545 <1> loc_find_longname:
2546 0000808B E87F210000 <1>      call    find_longname
2547 00008090 7328 <1>      jnc     short loc_print_longname
2548 <1>
2549 <1>      or      al, al
2550 00008094 741A <1>      jz      short loc_longname_not_found
2551 <1>
2552 <1>      ; 16/10/2016 (15h -> 15, 17)
2553 00008096 3C0F <1>      cmp     al, 15
2554 00008098 0F84C5F7FFFF <1>      je      cd_drive_not_ready ; drive not ready
2555 <1>      ; or
2556 0000809E 3C11 <1>      cmp     al, 17 ; read error
2557 000080A0 0F84BDF7FFFF <1>      je      cd_drive_not_ready
2558 <1>
2559 <1> loc_ln_file_dir_not_found:
2560 000080A6 BE[D3FD0000] <1>      mov     esi, Msg_File_Directory_Not_Found
2561 <1>      ;call    print_msg
2562 <1>      ;retn
2563 000080AB E90DE2FFFF <1>      jmp     print_msg
2564 <1>
2565 <1> loc_longname_not_found:
2566 000080B0 BE[F2FD0000] <1>      mov     esi, Msg_LongName_Not_Found
2567 <1>      ;call    print_msg
2568 <1>      ;retn
2569 000080B5 E903E2FFFF <1>      jmp     print_msg
2570 <1>
2571 <1> loc_print_longname:
2572 <1>      ;mov     esi, LongFileName
2573 000080BA BF[E6390100] <1>      mov     edi, TextBuffer
2574 000080BF 57 <1>      push    edi
2575 000080C0 3C00 <1>      cmp     al, 0
2576 000080C2 7708 <1>      ja      short loc_print_longname_1
2577 <1> loc_print_FS_longname: ; Singlix FS (64 byte ASCIIZ file name)
2578 000080C4 AC <1>      lodsb
2579 000080C5 AA <1>      stosb
2580 000080C6 08C0 <1>      or      al, al
2581 000080C8 75FA <1>      jnz     short loc_print_FS_longname
2582 000080CA EB07 <1>      jmp     short loc_print_longname_2
2583 <1>      ;
2584 <1> loc_print_longname_1: ; MS Windows long name (UNICODE chars)
2585 000080CC 66AD <1>      lodsw
2586 000080CE AA <1>      stosb
2587 000080CF 08C0 <1>      or      al, al
2588 000080D1 75F9 <1>      jnz     short loc_print_longname_1
2589 <1>      ;
2590 <1> loc_print_longname_2:
2591 000080D3 5E <1>      pop     esi
2592 000080D4 E8E4E1FFFF <1>      call    print_msg
2593 000080D9 BE[13070100] <1>      mov     esi, nextline
2594 <1>      ;call    print_msg
2595 <1>      ;retn
2596 000080DE 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 000080E3 BF[8A420100] <1> mov edi, FindFile_Drv
2607 000080E8 E879200000 <1> call parse_path_name
2608 000080ED 0F825AF9FFFF <1> jc loc_cmd_failed
2609 <1>
2610 <1> loc_show_check_filename_exists:
2611 000080F3 BE[CC420100] <1> mov esi, FindFile_Name
2612 000080F8 803E20 <1> cmp byte [esi], 20h
2613 000080FB 0F864CF9FFFF <1> jna loc_cmd_failed
2614 <1>
2615 <1> ; 15/02/2016 (invalid file name check)
2616 00008101 E809020000 <1> call check_filename
2617 00008106 730A <1> jnc short loc_show_change_drv
2618 <1>
2619 00008108 BE[BEFE0000] <1> mov esi, Msg_invalid_name_chars
2620 0000810D E9ABE1FFFF <1> jmp print_msg
2621 <1>
2622 <1> loc_show_change_drv:
2623 00008112 8A35[E6380100] <1> mov dh, [Current_Drv]
2624 00008118 8835[46410100] <1> mov [RUN_CDRV], dh
2625 0000811E 8A15[8A420100] <1> mov dl, [FindFile_Drv]
2626 00008124 38F2 <1> cmp dl, dh
2627 00008126 740B <1> je short loc_show_change_directory
2628 00008128 E890EAF9FFFF <1> call change_current_drive
2629 <1> ;jc loc_file_rw_cmd_failed
2630 0000812D 0F8245F9FFFF <1> jc loc_run_cmd_failed
2631 <1>
2632 <1> loc_show_change_directory:
2633 00008133 803D[8B420100]20 <1> cmp byte [FindFile_Directory], 20h
2634 0000813A 7618 <1> jna short loc_findload_showfile
2635 <1>
2636 0000813C FE05[A8FA0000] <1> inc byte [Restore_CDIR]
2637 00008142 BE[8B420100] <1> mov esi, FindFile_Directory
2638 00008147 30E4 <1> xor ah, ah ; CD_COMMAND sign -> 0
2639 00008149 E8041A0000 <1> call change_current_directory
2640 <1> ;jc loc_file_rw_cmd_failed
2641 0000814E 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 00008154 BE[CC420100] <1> mov esi, FindFile_Name
2649 00008159 BF[7B420100] <1> mov edi, Dir_Entry_Name ; Dir Entry Format File Name
2650 0000815E E8381F0000 <1> call convert_file_name
2651 00008163 89FE <1> mov esi, edi ; offset Dir_Entry_Name
2652 <1>
2653 00008165 28C0 <1> sub al, al ; Attrib AND mask = 0
2654 <1> ; Directory attribute : 10h
2655 <1> ; Volume name attribute: 8h
2656 00008167 B418 <1> mov ah, 00011000b ; 18h (Attrib NAND, AND --> zero mask)
2657 <1> ;
2658 00008169 6631C9 <1> xor cx, cx
2659 0000816C E8351C0000 <1> call locate_current_dir_file
2660 <1> ;jc loc_file_rw_cmd_failed
2661 00008171 0F8201F9FFFF <1> jc loc_run_cmd_failed
2662 <1>
2663 <1> loc_show_load_file:
2664 <1> ; EDI = Directory Entry
2665 00008177 668B4714 <1> mov ax, [edi+DirEntry_FstClusHI] ; First Cluster High Word
2666 0000817B C1E010 <1> shl eax, 16
2667 0000817E 668B471A <1> mov ax, [edi+DirEntry_FstClusLO] ; First Cluster Low Word
2668 00008182 A3[34430100] <1> mov [Show_Cluster], eax
2669 00008187 8B471C <1> mov eax, [edi+DirEntry_FileSize] ; File Size
2670 0000818A 21C0 <1> and eax, eax ; Empty file !
2671 0000818C 0F8491000000 <1> jz end_of_show_file
2672 00008192 A3[38430100] <1> mov [Show_FileSize], eax
2673 00008197 31C0 <1> xor eax, eax
2674 00008199 A3[3C430100] <1> mov [Show_FilePointer], eax ; 0
2675 0000819E 66A3[40430100] <1> mov [Show_ClusterPointer], ax ; 0
2676 000081A4 29DB <1> sub ebx, ebx
2677 000081A6 8A3D[E6380100] <1> mov bh, [Current_Drv]
2678 000081AC BE00010900 <1> mov esi, Logical_DOSDisks
2679 000081B1 01DE <1> add esi, ebx
2680 000081B3 8935[30430100] <1> mov [Show_LDDDT], esi ; Logical DOS Drv Description Table addr
2681 <1>
2682 000081B9 807E0300 <1> cmp byte [esi+LD_FATType], 0
2683 000081BD 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 000081BF 8B15[34430100] <1> mov edx, [Show_Cluster] ; Compatibility dir. buffer value (FDT)
2687 000081C5 8915[2C430100] <1> mov [Show_FDT], edx
2688 000081CB 31C0 <1> xor eax, eax
2689 000081CD A3[34430100] <1> mov [Show_Cluster], eax ; Sector index = 0
2690 <1> ; (next time it will be 1)
2691 <1> loc_show_calculate_cluster_size:
2692 000081D2 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 000081D6 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 000081D9 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 000081DB 66A3[42430100] <1> mov [Show_ClusterSize], ax
2701 <1>
2702 <1> loc_start_show_file:
2703 000081E1 BE[13070100] <1> mov esi, nextline
2704 000081E6 E8D2E0FFFF <1> call print_msg
2705 <1>
2706 000081EB A1[34430100] <1> mov eax, [Show_Cluster]
2707 000081F0 C605[44430100]17 <1> mov byte [Show_RowCount], 23

```

```

2708 <1>
2709 <1> ; 17/02/2016
2710 000081F7 8B35[30430100] <1> mov esi, [Show_LDDDT]
2711 <1>
2712 <1> loc_show_next_cluster:
2713 <1> ; 15/02/2016
2714 000081FD BB00000700 <1> mov ebx, Cluster_Buffer ; 70000h (for current TRDOS 386 version)
2715 <1> ; ESI = Logical DOS drv description table address
2716 00008202 E89A380000 <1> call read_cluster
2717 <1> ;jc loc_file_rw_cmd_failed
2718 00008207 0F826BF8FFFF <1> jc loc_run_cmd_failed
2719 <1>
2720 0000820D 31DB <1> xor ebx, ebx
2721 <1> loc_show_next_byte:
2722 0000820F 803D[44430100]00 <1> cmp byte [Show_RowCount], 0
2723 00008216 7521 <1> jne short pass_show_wait_for_key
2724 00008218 30E4 <1> xor ah, ah
2725 0000821A E8F789FFFF <1> call int16h
2726 0000821F 3C1B <1> cmp al, 1Bh
2727 00008221 750F <1> jne short pass_exit_show
2728 <1> end_of_show_file:
2729 <1> pass_show_file:
2730 00008223 BE[13070100] <1> mov esi, nextline
2731 00008228 E890E0FFFF <1> call print_msg
2732 0000822D E94D010000 <1> jmp loc_file_rw_restore_retn
2733 <1>
2734 <1> pass_exit_show:
2735 00008232 C605[44430100]14 <1> mov byte [Show_RowCount], 20
2736 <1> pass_show_wait_for_key:
2737 00008239 81C300000700 <1> add ebx, Cluster_Buffer
2738 0000823F 8A03 <1> mov al, [ebx]
2739 00008241 3C0D <1> cmp al, 0Dh
2740 00008243 0F8590000000 <1> jne loc_show_check_tab_space
2741 00008249 FE0D[44430100] <1> dec byte [Show_RowCount]
2742 <1> pass_show_dec_rowcount:
2743 0000824F B307 <1> mov bl, 7 ; (light gray character color, black background)
2744 00008251 8A3D[4E380100] <1> mov bh, [ACTIVE_PAGE] ; [ptty]
2745 00008257 E8569AFFFF <1> call _write_tty
2746 <1> loc_show_check_eof:
2747 0000825C FF05[3C430100] <1> inc dword [Show_FilePointer]
2748 00008262 A1[3C430100] <1> mov eax, [Show_FilePointer]
2749 00008267 3B05[38430100] <1> cmp eax, [Show_FileSize]
2750 0000826D 73B4 <1> jnb short end_of_show_file
2751 0000826F 66FF05[40430100] <1> inc word [Show_ClusterPointer]
2752 00008276 0FB71D[40430100] <1> movzx ebx, word [Show_ClusterPointer]
2753 <1>
2754 <1> ; 17/02/2016
2755 <1> ; (sector boundary -9 bits- check, 512 = 0)
2756 0000827D 66F7C3FF01 <1> test bx, 1FFh ; 1 to 511
2757 00008282 758B <1> jnz short loc_show_next_byte
2758 <1>
2759 <1> ; 16/02/2016
2760 00008284 8B35[30430100] <1> mov esi, [Show_LDDDT]
2761 <1> ;
2762 0000828A 807E0300 <1> cmp byte [esi+LD_FATType], 0
2763 0000828E 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 00008290 A1[34430100] <1> mov eax, [Show_Cluster]
2768 00008295 40 <1> inc eax
2769 00008296 A3[34430100] <1> mov [Show_Cluster], eax
2770 <1>
2771 0000829B 6621DB <1> and bx, bx ; 65536 -> 0
2772 0000829E 0F856BFFFFFF <1> jnz loc_show_next_byte
2773 000082A4 E954FFFFFF <1> jmp loc_show_next_cluster
2774 <1>
2775 <1> loc_show_check_fat_cluster_size:
2776 <1> ; 17/02/2016
2777 000082A9 663B1D[42430100] <1> cmp bx, [Show_ClusterSize] ; cluster size in bytes
2778 000082B0 0F8259FFFFFF <1> jb loc_show_next_byte
2779 000082B6 66C705[40430100]00- <1> mov word [Show_ClusterPointer], 0
2780 <1>
2781 000082BF A1[34430100] <1> mov eax, [Show_Cluster]
2782 <1> ;mov esi, [Show_LDDDT]
2783 <1> loc_show_get_next_cluster:
2784 000082C4 E8B4350000 <1> call get_next_cluster
2785 <1> ;jc loc_file_rw_cmd_failed
2786 000082C9 0F82A9F7FFFF <1> jc loc_run_cmd_failed
2787 <1> loc_show_update_ccluster:
2788 000082CF A3[34430100] <1> mov [Show_Cluster], eax
2789 000082D4 E924FFFFFF <1> jmp loc_show_next_cluster
2790 <1>
2791 <1> loc_show_check_tab_space:
2792 000082D9 3C09 <1> cmp al, 09h
2793 000082DB 0F856EFFFFFF <1> jne pass_show_dec_rowcount
2794 <1> loc_show_put_tab_space:
2795 000082E1 8A3D[4E380100] <1> mov bh, [ACTIVE_PAGE] ; [ptty]
2796 000082E7 E85596FFFF <1> call get_cpos
2797 <1> ; dl = cursor column
2798 000082EC 80E207 <1> and dl, 7 ; 18/02/2016
2799 <1> ;shr bh, 1 ; [ACTIVE_PAGE]
2800 000082EF 8A3D[4E380100] <1> mov bh, [ACTIVE_PAGE]
2801 000082F5 B307 <1> mov bl, 7 ; color attribute
2802 <1> loc_show_put_space_chars:
2803 000082F7 B020 <1> mov al, 20h ; space
2804 <1> ;mov bh, [ACTIVE_PAGE] ; [ptty]
2805 <1> ;mov bl, 7 ; color attribute
2806 000082F9 6652 <1> push dx
2807 000082FB E8B299FFFF <1> call _write_tty
2808 00008300 665A <1> pop dx

```

```

2809      <1>      ; 18/02/2016
2810      <1>      cmp     dl, 7
2811      <1>      jnb     loc_show_check_eof
2812      <1>      inc     dl
2813      <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 = ASCIIIZ 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      <1>      push     esi
2849      <1>
2850      <1>      mov     edi, invalid_fname_chars
2851      <1>      lodsb
2852      <1> check_filename_next_char:
2853      <1>      mov     ecx, sizeInvFnChars
2854      <1>      mov     edi, invalid_fname_chars
2855      <1> loc_scan_invalid_filename_char:
2856      <1>      scasb
2857      <1>      je      short loc_invalid_filename_stc
2858      <1>      loop   loc_scan_invalid_filename_char
2859      <1>      lodsb
2860      <1>      cmp     al, 1Fh ; 20h and above
2861      <1>      ja      short check_filename_next_char
2862      <1>
2863      <1> check_filename_dot:
2864      <1>      mov     esi, [esp]
2865      <1>
2866      <1>      mov     ah, 21h
2867      <1>      mov     ecx, 8
2868      <1> loc_check_filename_next_char:
2869      <1>      lodsb
2870      <1>      cmp     al, 2Eh
2871      <1>      jne     short pass_check_fn_dot_check
2872      <1> loc_check_filename_ext_0:
2873      <1>      lodsb
2874      <1>      cmp     al, ah ; 21h
2875      <1>      jb      short loc_invalid_filename
2876      <1>      cmp     al, 2Eh
2877      <1>      jne     short loc_check_filename_ext_1
2878      <1>
2879      <1> loc_invalid_filename_stc:
2880      <1> loc_check_fn_stc_rtn:
2881      <1>      stc
2882      <1> loc_invalid_filename:
2883      <1>      ; 10/10/2016 (0Bh -> 26)
2884      <1>      mov     eax, ERR_INV_FILE_NAME ; (=26)
2885      <1>      ; Invalid file name chars
2886      <1> loc_check_fn_rtn:
2887      <1>      pop     esi
2888      <1>      retn
2889      <1>
2890      <1> pass_check_fn_dot_check:
2891      <1>      cmp     al, ah ; 21h
2892      <1>      jb      short loc_check_fn_clc_rtn
2893      <1>      loop   loc_check_filename_next_char
2894      <1>      lodsb
2895      <1>      cmp     al, ah ; 21h
2896      <1>      jb      short loc_check_fn_clc_rtn
2897      <1>      cmp     al, 2Eh
2898      <1>      jne     short loc_check_fn_stc_rtn
2899      <1>      jmp     short loc_check_filename_ext_0
2900      <1>
2901      <1> loc_check_filename_ext_1:
2902      <1>      lodsb
2903      <1>      cmp     al, ah ; 21h
2904      <1>      jb      short loc_check_fn_clc_rtn
2905      <1>      cmp     al, 2Eh
2906      <1>      je      short loc_check_fn_stc_rtn
2907      <1>      lodsb
2908      <1>      cmp     al, ah ; 21h
2909      <1>      jb      short loc_check_fn_clc_rtn
2910      <1>      cmp     al, 2Eh

```


2911	0000836B	74D5	<1>	je	short loc_check_fn_stc_rtn
2912	0000836D	AC	<1>	lodsb	
2913	0000836E	38E0	<1>	cmp	al, ah ; 21h
2914	00008370	73D0	<1>	jnb	short loc_check_fn_stc_rtn
2915			<1>		
2916			<1>	loc_check_fn_clc_rtn:	
2917	00008372	5E	<1>	pop	esi
2918	00008373	F8	<1>	clc	
2919	00008374	C3	<1>	retn	
2920			<1>		
2921			<1>	loc_print_deleted_message:	
2922	00008375	BE[93FF0000]	<1>	mov	esi, Msg_Deleted
2923	0000837A	E83EDFFFFFFF	<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	0000837F	9C	<1>	pushf	
2932	00008380	E84DF7FFFF	<1>	call	restore_cdir_after_cmd_fail
2933	00008385	9D	<1>	popf	
2934	00008386	720D	<1>	jc	short loc_file_rw_check_write_fault
2935	00008388	C3	<1>	retn	
2936			<1>		
2937			<1>	loc_permission_denied:	
2938			<1>	; 27/02/2016	
2939	00008389	BE[A0FF0000]	<1>	mov	esi, Msg_Permission_Denied
2940	0000838E	E82ADFFFFFFF	<1>	call	print_msg
2941	00008393	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	00008395	3C12	<1>	cmp	al, 18 ; 05/11/2016
2946	00008397	0F85E0F6FFFF	<1>	jne	loc_run_cmd_failed_cmp_al
2947	0000839D	BE[88FD0000]	<1>	mov	esi, Msg_Not_Ready_Write_Err
2948			<1>	;call print_msg	
2949			<1>	;retn	
2950	000083A2	E916DFFFFFFF	<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>	;	
2959			<1>	get_mkdir_fchar:	
2960			<1>	; esi = directory name	
2961	000083A7	803E20	<1>	cmp	byte [esi], 20h
2962	000083AA	7701	<1>	ja	short loc_mkdir_parse_path_name
2963			<1>		
2964			<1>	loc_mkdir_nodirname_retn:	
2965	000083AC	C3	<1>	retn	
2966			<1>		
2967			<1>	loc_mkdir_parse_path_name:	
2968	000083AD	BF[8A420100]	<1>	mov	edi, FindFile_Drv
2969	000083B2	E8AF1D0000	<1>	call	parse_path_name
2970	000083B7	0F8290F6FFFF	<1>	jc	loc_cmd_failed
2971			<1>		
2972			<1>	loc_mkdir_check_dirname_exists:	
2973	000083BD	BE[CC420100]	<1>	mov	esi, FindFile_Name
2974	000083C2	803E20	<1>	cmp	byte [esi], 20h
2975	000083C5	0F8682F6FFFF	<1>	jna	loc_cmd_failed
2976	000083CB	8935[48430100]	<1>	mov	[DelFile_FNPointer], esi
2977	000083D1	E839FFFFFFF	<1>	call	check_filename
2978	000083D6	7259	<1>	jc	short loc_mkdir_invalid_dir_name_chars
2979			<1>		
2980			<1>	loc_mkdir_drv:	
2981	000083D8	8A35[E6380100]	<1>	mov	dh, [Current_Drv]
2982	000083DE	8835[46410100]	<1>	mov	[RUN_CDRV], dh
2983			<1>		
2984	000083E4	8A15[8A420100]	<1>	mov	dl, [FindFile_Drv]
2985	000083EA	38F2	<1>	cmp	dl, dh
2986	000083EC	7407	<1>	je	short loc_mkdir_change_directory

```

3013 00008422 BE[EBFE0000]      <1>      mov     esi, Msg_Name_Exists
3014 00008427 E891DEFFFF      <1>      call    print_msg
3015                                <1>
3016 0000842C E94EFFFFFF      <1>      jmp     loc_file_rw_restore_retn
3017                                <1>
3018                                <1> loc_mkdir_invalid_dir_name_chars:
3019 00008431 BE[BEFE0000]      <1>      mov     esi, Msg_invalid_name_chars
3020 00008436 E882DEFFFF      <1>      call    print_msg
3021                                <1>
3022 0000843B E93FFFFFFF      <1>      jmp     loc_file_rw_restore_retn
3023                                <1>
3024                                <1> loc_mkdir_check_error_code:
3025 00008440 3C02              <1>      cmp     al, 2
3026                                <1>      ;je     short loc_mkdir_directory_not_found
3027 00008442 7406              <1>      je      short loc_mkdir_ask_for_yes_no
3028 00008444 F9                      <1>      stc
3029 00008445 E935FFFFFF      <1>      jmp     loc_file_rw_cmd_failed
3030                                <1>
3031                                <1> loc_mkdir_directory_not_found:
3032                                <1> loc_mkdir_ask_for_yes_no:
3033 0000844A BE[0CFF0000]      <1>      mov     esi, Msg_DoYouWantMkdir
3034 0000844F E869DEFFFF      <1>      call    print_msg
3035 00008454 8B35[48430100]    <1>      mov     esi, [DelFile_FNPointer]
3036 0000845A E85EDEFFFF      <1>      call    print_msg
3037 0000845F BE[2BFF0000]      <1>      mov     esi, Msg_YesNo
3038 00008464 E854DEFFFF      <1>      call    print_msg
3039                                <1>
3040 00008469 C605[35FF0000]20  <1>      mov     byte [Y_N_nextline], 20h
3041                                <1>
3042                                <1> loc_mkdir_ask_again:
3043 00008470 30E4              <1>      xor     ah, ah
3044 00008472 E89F87FFFF      <1>      call    int16h
3045 00008477 3C1B              <1>      cmp     al, 1Bh
3046                                <1>      ;je     short loc_do_not_make_directory
3047 00008479 7447              <1>      je      short loc_mkdir_y_n_escape
3048 0000847B 24DF              <1>      and     al, 0DFh ; y -> Y, n -> N
3049 0000847D 3C59              <1>      cmp     al, 'Y' ; 'yes'
3050 0000847F 7404              <1>      je      short loc_mkdir_yes_make_directory
3051 00008481 3C4E              <1>      cmp     al, 'N' ; 'no'
3052 00008483 75EB              <1>      jne     short loc_mkdir_ask_again
3053                                <1>
3054                                <1> loc_do_not_make_directory:
3055                                <1> loc_mkdir_yes_make_directory:
3056 00008485 A2[35FF0000]      <1>      mov     [Y_N_nextline], al
3057 0000848A 6650              <1>      push    ax
3058 0000848C BE[35FF0000]      <1>      mov     esi, Y_N_nextline
3059 00008491 E827DEFFFF      <1>      call    print_msg
3060 00008496 6658              <1>      pop     ax
3061                                <1>      ;cmp    al, 'Y' ; 'yes'
3062                                <1>      ;cmc
3063                                <1>      ;jnc   loc_file_rw_restore_retn
3064 00008498 3C4E              <1>      cmp     al, 'N' ; 'no'
3065 0000849A 0F84DFEFFFFF      <1>      je      loc_file_rw_restore_retn
3066                                <1>
3067                                <1> loc_mkdir_call_make_sub_directory:
3068 000084A0 8B35[48430100]    <1>      mov     esi, [DelFile_FNPointer]
3069 000084A6 B110              <1>      mov     cl, 10h ; Directory attributes
3070 000084A8 E8B81D0000      <1>      call    make_sub_directory
3071                                <1> loc_rename_file_ok: ; 06/03/2016
3072 000084AD 0F82CCFEFFFF      <1>      jc      loc_file_rw_cmd_failed
3073                                <1> move_source_file_to_destination_OK:
3074 000084B3 BE[39FF0000]      <1>      mov     esi, Msg_OK
3075 000084B8 E800DEFFFF      <1>      call    print_msg
3076 000084BD E9BDFEFFFF      <1>      jmp     loc_file_rw_restore_retn
3077                                <1>
3078                                <1> loc_mkdir_y_n_escape:
3079 000084C2 B04E              <1>      mov     al, 'N' ; 'no'
3080 000084C4 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 000084C6 803E20      <1>      cmp     byte [esi], 20h
3096 000084C9 7701              <1>      ja      short loc_rmdir_parse_path_name
3097                                <1>
3098                                <1> loc_rmdir_nodirname_retn:
3099 000084CB C3                      <1>      retn
3100                                <1>
3101                                <1> loc_rmdir_parse_path_name:
3102 000084CC BF[8A420100]    <1>      mov     edi, FindFile_Drv
3103 000084D1 E8901C0000      <1>      call    parse_path_name
3104 000084D6 0F8271F5FFFF      <1>      jc      loc_cmd_failed
3105                                <1>
3106                                <1> loc_rmdir_check_dirname_exists:
3107 000084DC BE[CC420100]    <1>      mov     esi, FindFile_Name
3108 000084E1 803E20      <1>      cmp     byte [esi], 20h
3109 000084E4 0F8663F5FFFF      <1>      jna     loc_cmd_failed
3110 000084EA 8935[48430100]    <1>      mov     [DelFile_FNPointer], esi
3111                                <1>
3112                                <1> loc_rmdir_drv:
3113 000084F0 8A35[E6380100]    <1>      mov     dh, [Current_Drv]
3114 000084F6 8835[46410100]    <1>      mov     [RUN_CDRV], dh

```

```

3115                                     <1>
3116 000084FC 8A15[8A420100]          <1>      mov     dl, [FindFile_Drv]
3117 00008502 38F2                    <1>      cmp     dl, dh
3118 00008504 740B                    <1>      je      short loc_rmdir_change_directory
3119                                     <1>
3120 00008506 E8B2E6FFFF                <1>      call    change_current_drive
3121 0000850B 0F826EFFFFFF              <1>      jc      loc_file_rw_cmd_failed
3122                                     <1>
3123                                     <1> loc_rmdir_change_directory:
3124 00008511 803D[8B420100]20          <1>      cmp     byte [FindFile_Directory], 20h
3125 00008518 7614                    <1>      jna     short loc_rmdir_find_directory
3126                                     <1>
3127 0000851A FE05[A8FA0000]            <1>      inc     byte [Restore_CDIRE]
3128 00008520 BE[8B420100]              <1>      mov     esi, FindFile_Directory
3129 00008525 30E4                    <1>      xor     ah, ah ; CD_COMMAND sign -> 0
3130 00008527 E826160000                <1>      call    change_current_directory
3131 0000852C 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 0000852E 8B35[48430100]            <1>      mov     esi, [DelFile_FNPointer]
3139 00008534 66B81008                  <1>      mov     ax, 0810h ; Only directories
3140 00008538 E812FAFFFF                <1>      call    find_first_file
3141 0000853D 730A                    <1>      jnc     short loc_rmdir_ambgfn_check
3142                                     <1>
3143                                     <1> loc_rmdir_check_error_code:
3144 0000853F 3C02                    <1>      cmp     al, 2
3145 00008541 740B                    <1>      je      short loc_rmdir_directory_not_found
3146 00008543 F9                      <1>      stc
3147 00008544 E936FEFFFF                <1>      jmp     loc_file_rw_cmd_failed
3148                                     <1>
3149                                     <1> loc_rmdir_ambgfn_check:
3150 00008549 6621D2                    <1>      and     dx, dx ; Ambiguous filename chars used sign (DX>0)
3151 0000854C 740F                    <1>      jz      short loc_rmdir_directory_found
3152                                     <1>
3153                                     <1> loc_rmdir_directory_not_found:
3154 0000854E BE[AAFD0000]              <1>      mov     esi, Msg_Dir_Not_Found
3155 00008553 E865DDFFFF                <1>      call    print_msg
3156                                     <1>
3157 00008558 E922FEFFFF                <1>      jmp     loc_file_rw_restore_retn
3158                                     <1>
3159                                     <1> loc_rmdir_directory_found:
3160 0000855D 80E307                    <1>      and     bl, 07h ; Attributes
3161 00008560 0F8523FEFFFF              <1>      jnz     loc_permission_denied
3162                                     <1>
3163                                     <1> loc_rmdir_save_lnel: ; 28/02/2016
3164                                     <1>      ;mov     bh, [LongName_EntryLength]
3165 00008566 883D[52430100]            <1>      mov     [DelFile_LNEL], bh ; Long name entry length (if > 0)
3166                                     <1>      ; edi = Directory Entry Offset (DirBuff)
3167                                     <1>      ; esi = Directory Entry (FFF Structure)
3168                                     <1>      ;mov     [DelFile_DirEntryAddr], edi ; not required
3169                                     <1>      ;mov     ax, [edi+20] ; First Cluster High Word
3170                                     <1>      ;shl     eax, 16
3171                                     <1>      ;mov     ax, [edi+26] ; First Cluster Low Word
3172                                     <1>      ; ROOT Dir First Cluster = 0
3173                                     <1>      ;cmp     eax, 2
3174                                     <1>      ;jbe     loc_update_direntry_1
3175                                     <1>
3176                                     <1> pass_rmdir_fc_check:
3177 0000856C 57                      <1>      push     edi ; * (29/02/2016)
3178                                     <1>
3179 0000856D BE[3FFF0000]              <1>      mov     esi, Msg_DoYouWantRmdir
3180 00008572 E846DDFFFF                <1>      call    print_msg
3181 00008577 8B35[48430100]            <1>      mov     esi, [DelFile_FNPointer]
3182 0000857D E83BDDFFFF                <1>      call    print_msg
3183 00008582 BE[2BFF0000]              <1>      mov     esi, Msg_YesNo
3184 00008587 E831DDFFFF                <1>      call    print_msg
3185                                     <1>
3186                                     <1> loc_rmdir_ask_again:
3187 0000858C 30E4                    <1>      xor     ah, ah
3188 0000858E E88386FFFF                <1>      call    int16h
3189 00008593 3C1B                    <1>      cmp     al, 1Bh
3190                                     <1>      ;je      short loc_do_not_delete_directory
3191 00008595 0F8498000000              <1>      je      loc_rmdir_y_n_escape ; 06/03/2016
3192 0000859B 24DF                    <1>      and     al, 0DFh
3193 0000859D A2[35FF0000]              <1>      mov     [Y_N_nextline], al
3194 000085A2 3C59                    <1>      cmp     al, 'Y'
3195 000085A4 7404                    <1>      je      short loc_rmdir_yes_delete_directory
3196 000085A6 3C4E                    <1>      cmp     al, 'N'
3197 000085A8 75E2                    <1>      jne     short loc_rmdir_ask_again
3198                                     <1>
3199                                     <1> loc_do_not_delete_directory:
3200                                     <1> loc_rmdir_yes_delete_directory:
3201 000085AA A2[35FF0000]              <1>      mov     [Y_N_nextline], al
3202 000085AF 6650                    <1>      push     ax
3203 000085B1 BE[35FF0000]              <1>      mov     esi, Y_N_nextline
3204 000085B6 E802DDFFFF                <1>      call    print_msg
3205 000085BB 6658                    <1>      pop      ax
3206 000085BD 5F                      <1>      pop     edi ; * (29/02/2016)
3207                                     <1>      ;cmp     al, 'Y' ; 'yes'
3208                                     <1>      ;cmc
3209                                     <1>      ;jnc     loc_file_rw_restore_retn
3210 000085BE 3C4E                    <1>      cmp     al, 'N' ; 'no'
3211 000085C0 0F84B9FDFFFF              <1>      je      loc_file_rw_restore_retn
3212                                     <1>
3213                                     <1> loc_rmdir_delete_short_name_check_dir_empty:
3214                                     <1>      ; EDI = Directory buffer entry offset/address
3215 000085C6 668B4714                  <1>      mov     ax, [edi+20] ; First Cluster High Word
3216 000085CA C1E010                    <1>      shl     eax, 16

```

```

3217 000085CD 668B471A      <1>      mov     ax, [edi+26] ; First Cluster Low Word
3218                                <1>
3219 000085D1 A3[4C430100]    <1>      mov     [DelFile_FCluster], eax
3220                                <1>
3221                                <1>      ;mov     bx, [DirBuff_EntryCounter]
3222 000085D6 668B1D[04430100] <1>      mov     bx, [FindFile_DirEntryNumber] ; 27/02/2016
3223 000085DD 66891D[50430100] <1>      mov     [DelFile_EntryCounter], bx
3224                                <1>
3225 000085E4 29DB            <1>      sub     ebx, ebx
3226 000085E6 8A3D[8A420100]    <1>      mov     bh, [FindFile_Drv]
3227 000085EC BE00010900        <1>      mov     esi, Logical_DOSDisks
3228 000085F1 01DE            <1>      add     esi, ebx
3229                                <1>
3230 000085F3 66817F0CA101      <1>      cmp     word [edi+DirEntry_NTRes], 01A1h
3231 000085F9 743F            <1>      je      short loc_rmdir_delete_fs_directory
3232                                <1>
3233                                <1>      ;cmp     byte [esi+LD_FATType], 1
3234                                <1>      ;jnb     short loc_rmdir_get__last_cluster_0
3235                                <1>      ;mov     eax, 0Bh ; Invalid Format
3236                                <1>      ;jmp     loc_file_rw_cmd_failed
3237                                <1>
3238                                <1> ;loc_rmdir_get_last_cluster_0:
3239 000085FB 8B15[15410100]    <1>      mov     edx, [DirBuff_Cluster]
3240 00008601 8915[7C430100]    <1>      mov     [Rmdir_ParentDirCluster], edx
3241                                <1>
3242 00008607 893D[78430100]    <1>      mov     [Rmdir_DirEntryOffset], edi
3243                                <1>
3244                                <1>      ; 01/03/2016
3245 0000860D C705[06410100]0000- <1>      mov     dword [FAT_ClusterCounter], 0 ; Reset
3246 00008615 0000            <1>
3247                                <1> loc_rmdir_get_last_cluster:
3248 00008617 E85C3A0000        <1>      call    get_last_cluster
3249 0000861C 0F82B8000000        <1>      jc      loc_rmdir_cmd_failed
3250                                <1>
3251 00008622 3B05[4C430100]    <1>      cmp     eax, [DelFile_FCluster]
3252 00008628 752F            <1>      jne     short loc_rmdir_multi_dir_clusters
3253                                <1>
3254 0000862A C605[77430100]00    <1>      mov     byte [Rmdir_MultiClusters], 0
3255 00008631 EB2D            <1>      jmp     short pass_rmdir_multi_dir_clusters
3256                                <1>
3257                                <1> loc_rmdir_y_n_escape:
3258 00008633 B04E            <1>      mov     al, 'N' ; 'no'
3259 00008635 E970FFFFFF        <1>      jmp     loc_do_not_delete_directory
3260                                <1>
3261                                <1> loc_rmdir_delete_fs_directory:
3262 0000863A 807E04A1        <1>      cmp     byte [esi+LD_FSType], 0A1h
3263 0000863E 0F8545FDFFFF        <1>      jne     loc_permission_denied
3264                                <1>
3265 00008644 E826140000        <1>      call    delete_fs_directory
3266 00008649 0F8326FDFFFF        <1>      jnc     loc_print_deleted_message
3267                                <1>
3268 0000864F 09C0            <1>      or      eax, eax
3269 00008651 745D            <1>      jz      loc_rmdir_directory_not_empty_2
3270 00008653 F9                <1>      stc
3271 00008654 E926FDFFFF        <1>      jmp     loc_file_rw_cmd_failed
3272                                <1>
3273                                <1> loc_rmdir_multi_dir_clusters:
3274 00008659 C605[77430100]01    <1>      mov     byte [Rmdir_MultiClusters], 1
3275                                <1>
3276                                <1> pass_rmdir_multi_dir_clusters:
3277 00008660 A3[80430100]        <1>      mov     [Rmdir_DirLastCluster], eax
3278 00008665 890D[84430100]    <1>      mov     [Rmdir_PreviousCluster], ecx
3279                                <1>
3280                                <1> loc_rmdir_load_fat_sub_directory:
3281 0000866B E8F3330000        <1>      call    load_FAT_sub_directory
3282 00008670 7268            <1>      jc      loc_rmdir_cmd_failed
3283                                <1>
3284                                <1> loc_rmdir_find_last_dir_entry:
3285 00008672 56                <1>      push    esi
3286 00008673 BE[6E420100]    <1>      mov     esi, Dir_File_Name
3287 00008678 C6062A        <1>      mov     byte [esi], '*'
3288 0000867B C646082A        <1>      mov     byte [esi+8], '*'
3289 0000867F 31DB            <1>      xor     ebx, ebx ; Entry offset = 0
3290                                <1> loc_rmdir_find_last_dir_entry_next:
3291 00008681 66B80008        <1>      mov     ax, 0800h ; Except volume/long names
3292 00008685 6631C9        <1>      xor     cx, cx ; 0 = Find a valid file or dir name
3293 00008688 E81B180000        <1>      call    find_directory_entry
3294 0000868D 7271            <1>      jc      short loc_rmdir_empty_dir_cluster
3295 0000868F 83FB01        <1>      cmp     ebx, 1
3296 00008692 771B            <1>      ja      short loc_rmdir_directory_not_empty_1
3297                                <1> loc_rmdir_dot_entry_check:
3298 00008694 80FD2E        <1>      cmp     ch, '.' ; The first char of the dir entry
3299 00008697 7516            <1>      jne     short loc_rmdir_directory_not_empty_1
3300 00008699 08DB            <1>      or      bl, bl
3301 0000869B 7506            <1>      jnz     short loc_rmdir_dotdot_entry_check
3302 0000869D 807F0120        <1>      cmp     byte [edi+1], 20h
3303 000086A1 EB06            <1>      jmp     short pass_rmdir_dot_entry_check
3304                                <1>
3305                                <1> loc_rmdir_dotdot_entry_check:
3306 000086A3 66817F012E20      <1>      cmp     word [edi+1], '. '
3307                                <1> pass_rmdir_dot_entry_check:
3308 000086A9 7504            <1>      jne     short loc_rmdir_directory_not_empty_1
3309 000086AB FEC3            <1>      inc     bl
3310 000086AD EBD2            <1>      jmp     short loc_rmdir_find_last_dir_entry_next
3311                                <1>
3312                                <1>
3313                                <1> loc_rmdir_directory_not_empty_1:
3314 000086AF 58                <1>      pop     eax ; pushed esi
3315                                <1>
3316                                <1> loc_rmdir_directory_not_empty_2:
3317 000086B0 BE[60FF0000]    <1>      mov     esi, Msg_Dir_Not_Empty

```



```

3318 000086B5 E803DCFFFF <1> call print_msg
3319 <1> ; 01/03/2016
3320 000086BA A1[06410100] <1> mov eax, [FAT_ClusterCounter]
3321 000086BF 09C0 <1> or eax, eax ; 0 ?
3322 000086C1 0F84B8FCFFFF <1> jz loc_file_rw_restore_retn
3323 <1> ; ESI = Logical DOS Drive Description Table address
3324 <1>
3325 000086C7 66BB01FF <1> mov bx, 0FF01h ; BH = FFh -> use ESI for Drive parameters
3326 <1> ; BL = 1 -> add free clusters
3327 000086CB E829380000 <1> call calculate_fat_freespace
3328 000086D0 09C9 <1> or ecx, ecx
3329 000086D2 0F84A7FCFFFF <1> jz loc_file_rw_restore_retn ; ecx = 0 -> OK
3330 <1> ; ecx > 0 -> Error (Recalculation is needed)
3331 000086D8 EB0E <1> jmp short loc_rmdir_cmd_return
3332 <1>
3333 <1>
3334 <1> loc_rmdir_cmd_failed:
3335 000086DA 833D[06410100]01 <1> cmp dword [FAT_ClusterCounter], 1
3336 000086E1 0F8298FCFFFF <1> jb loc_file_rw_cmd_failed
3337 000086E7 F9 <1> stc
3338 <1> loc_rmdir_cmd_return:
3339 <1> ; 01/03/2016
3340 000086E8 9C <1> pushf
3341 <1> ; ESI = Logical DOS Drive Description Table address
3342 000086E9 66BB00FF <1> mov bx, 0FF00h ; BH = FFh -> use ESI for Drive parameters
3343 <1> ; BL = 0 -> Recalculate free cluster count
3344 000086ED 50 <1> push eax
3345 000086EE E806380000 <1> call calculate_fat_freespace
3346 000086F3 58 <1> pop eax
3347 000086F4 9D <1> popf
3348 000086F5 0F8284FCFFFF <1> jc loc_file_rw_cmd_failed
3349 000086FB E97FFCFFFF <1> jmp loc_file_rw_restore_retn
3350 <1>
3351 <1>
3352 <1> loc_rmdir_empty_dir_cluster:
3353 00008700 5E <1> pop esi
3354 <1>
3355 <1> loc_rmdir_set_prev_cluster_dir_last_cluster:
3356 00008701 803D[77430100]00 <1> cmp byte [RmDir_MultiClusters], 0
3357 00008708 761D <1> jna short loc_rmdir_unlink_dir_last_cluster
3358 <1>
3359 0000870A A1[84430100] <1> mov eax, [RmDir_PreviousCluster]
3360 <1> ;xor ecx, ecx
3361 0000870F 49 <1> dec ecx ; FFFFFFFFh
3362 00008710 E892340000 <1> call update_cluster
3363 00008715 7310 <1> jnc short loc_rmdir_unlink_dir_last_cluster
3364 <1>
3365 <1> loc_rmdir_unlink_stc_retn:
3366 <1> ; 01/03/2016
3367 00008717 83F801 <1> cmp eax, 1 ; eax = 0 -> end of cluster chain
3368 0000871A F5 <1> cmc
3369 0000871B 72BD <1> jc short loc_rmdir_cmd_failed
3370 0000871D EB1D <1> jmp short loc_rmdir_save_fat_buffer
3371 <1>
3372 <1> loc_rmdir_unlink_stc_retn_0Bh:
3373 0000871F B81C000000 <1> mov eax, 28 ; Invalid format ; 15/10/2016
3374 00008724 F9 <1> stc
3375 00008725 EBB3 <1> jmp short loc_rmdir_cmd_failed
3376 <1>
3377 <1> loc_rmdir_unlink_dir_last_cluster:
3378 00008727 A1[80430100] <1> mov eax, [RmDir_DirLastCluster]
3379 0000872C 31C9 <1> xor ecx, ecx ; 0
3380 0000872E E874340000 <1> call update_cluster
3381 00008733 73EA <1> jnc short loc_rmdir_unlink_stc_retn_0Bh
3382 <1> ; Because of it is the last cluster
3383 <1> ; 'update_cluster' must return with eocc error
3384 00008735 09C0 <1> or eax, eax
3385 00008737 7403 <1> jz short loc_rmdir_save_fat_buffer ; eocc
3386 00008739 F9 <1> stc
3387 0000873A EB9E <1> jmp short loc_rmdir_cmd_failed
3388 <1>
3389 <1> loc_rmdir_save_fat_buffer:
3390 0000873C 803D[FE400100]02 <1> cmp byte [FAT_BuffValidData], 2
3391 00008743 7525 <1> jne short loc_rmdir_calculate_FAT_freespace
3392 00008745 E81A370000 <1> call save_fat_buffer
3393 0000874A 728E <1> jc short loc_rmdir_cmd_failed
3394 <1>
3395 <1> ; 01/03/2016
3396 0000874C 803D[77430100]00 <1> cmp byte [RmDir_MultiClusters], 0
3397 00008753 7615 <1> jna short loc_rmdir_calculate_FAT_freespace
3398 <1>
3399 00008755 A1[4C430100] <1> mov eax, [DelFile_FCluster]
3400 0000875A E9B8FEFFFF <1> jmp loc_rmdir_get_last_cluster
3401 <1>
3402 <1> loc_rmdir_delete_short_name_invalid_data:
3403 0000875F B81D000000 <1> mov eax, 29 ; Invalid data (15/10/2016)
3404 00008764 F9 <1> stc
3405 00008765 E970FFFFFF <1> jmp loc_rmdir_cmd_failed
3406 <1>
3407 <1> loc_rmdir_calculate_FAT_freespace:
3408 0000876A A1[06410100] <1> mov eax, [FAT_ClusterCounter]
3409 0000876F 66BB01FF <1> mov bx, 0FF01h
3410 <1> ; BL = 1 -> Add EAX to free space count
3411 <1> ; BH = FFh ->
3412 <1> ; ESI = Logical DOS Drive Description Table address
3413 00008773 E881370000 <1> call calculate_fat_freespace
3414 <1>
3415 00008778 21C9 <1> and ecx, ecx ; ecx = 0 -> valid free sector count
3416 0000877A 7409 <1> jz short loc_rmdir_delete_short_name_continue
3417 <1>
3418 <1> loc_rmdir_recalculate_FAT_freespace:
3419 0000877C 66BB00FF <1> mov bx, 0FF00h ; BL = 0 -> Recalculate free space

```

```

3420 00008780 E874370000 <1> call calculate_fat_freespace
3421 <1>
3422 <1> loc_rmdir_delete_short_name_continue:
3423 00008785 A1[7C430100] <1> mov eax, [Rmdir_ParentDirCluster]
3424 0000878A 83F802 <1> cmp eax, 2
3425 0000878D 730D <1> jnb short loc_rmdir_del_short_name_load_sub_dir
3426 0000878F E844320000 <1> call load_FAT_root_directory
3427 00008794 0F82E5FBFFFF <1> jc loc_file_rw_cmd_failed
3428 0000879A EB0B <1> jmp short loc_rmdir_del_short_name_ld_chk_fclust
3429 <1>
3430 <1> loc_rmdir_del_short_name_load_sub_dir:
3431 0000879C E8C2320000 <1> call load_FAT_sub_directory
3432 000087A1 0F82D8FBFFFF <1> jc loc_file_rw_cmd_failed
3433 <1>
3434 <1> loc_rmdir_del_short_name_ld_chk_fclust:
3435 000087A7 0FB73D[78430100] <1> movzx edi, word [Rmdir_DirEntryOffset]
3436 000087AE 81C700000800 <1> add edi, Directory_Buffer
3437 <1>
3438 000087B4 668B4714 <1> mov ax, [edi+20] ; First Cluster High Word
3439 000087B8 C1E010 <1> shl eax, 16
3440 000087BB 668B471A <1> mov ax, [edi+26] ; First Cluster Low Word
3441 <1> ; Not necessary...
3442 000087BF 3B05[4C430100] <1> cmp eax, [DelFile_FCluster]
3443 000087C5 7598 <1> jne short loc_rmdir_delete_short_name_invalid_data
3444 <1> ;
3445 000087C7 C607E5 <1> mov byte [edi], 0E5h ; 'Deleted' sign
3446 <1> ; 27/02/2016
3447 <1> ; TRDOS v1 has a bug here! it does not set
3448 <1> ; 'DirBuff_ValidData' to 2; as result of this bug,
3449 <1> ; 'save_directory_buffer' would not save the change !
3450 000087CA C605[10410100]02 <1> mov byte [DirBuff_ValidData], 2 ; change sign
3451 <1> ;
3452 000087D1 E8F41D0000 <1> call save_directory_buffer
3453 000087D6 0F82A3FBFFFF <1> jc loc_file_rw_cmd_failed
3454 <1>
3455 <1> loc_rmdir_del_long_name:
3456 000087DC 0FB615[52430100] <1> movzx edx, byte [DelFile_LNEL]
3457 000087E3 08D2 <1> or dl, dl
3458 000087E5 7414 <1> jz short loc_rmdir_update_parent_dir_lmdt
3459 <1>
3460 000087E7 0FB705[50430100] <1> movzx eax, word [DelFile_EntryCounter]
3461 000087EE 29D0 <1> sub eax, edx
3462 000087F0 0F8289FBFFFF <1> jc loc_file_rw_cmd_failed
3463 <1>
3464 <1> ; EAX = Directory Entry Number of the long name last entry
3465 000087F6 E82F1F0000 <1> call delete_longname
3466 <1> ;jc short loc_file_rw_cmd_failed
3467 <1>
3468 <1> loc_rmdir_update_parent_dir_lmdt:
3469 000087FB E8651E0000 <1> call update_parent_dir_lmdt
3470 <1> ;jc short loc_file_rw_cmd_failed
3471 <1>
3472 <1> loc_rmdir_ok:
3473 00008800 BE[39FF0000] <1> mov esi, Msg_OK
3474 00008805 E8B3DAFFFF <1> call print_msg
3475 0000880A E970FBFFFF <1> jmp loc_file_rw_restore_retn
3476 <1>
3477 <1>
3478 <1> delete_file:
3479 <1> ; 29/02/2016
3480 <1> ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
3481 <1> ; 09/08/2010 (CMD_INTR.ASM, 'cmp_cmd_del')
3482 <1> ; 28/02/2010
3483 <1>
3484 <1> get_delfile_fchar:
3485 <1> ; esi = file name
3486 0000880F 803E20 <1> cmp byte [esi], 20h
3487 00008812 7701 <1> ja short loc_delfile_parse_path_name
3488 <1>
3489 <1> loc_delfile_nofilename_retn:
3490 00008814 C3 <1> retn
3491 <1>
3492 <1> loc_delfile_parse_path_name:
3493 00008815 BF[8A420100] <1> mov edi, FindFile_Drv
3494 0000881A E847190000 <1> call parse_path_name
3495 0000881F 0F8228F2FFFF <1> jc loc_cmd_failed
3496 <1>
3497 <1> loc_delfile_check_filename_exists:
3498 00008825 BE[CC420100] <1> mov esi, FindFile_Name
3499 0000882A 803E20 <1> cmp byte [esi], 20h
3500 0000882D 0F861AF2FFFF <1> jna loc_cmd_failed
3501 00008833 8935[48430100] <1> mov [DelFile_FNPointer], esi
3502 <1>
3503 <1> loc_delfile_drv:
3504 00008839 8A15[8A420100] <1> mov dl, [FindFile_Drv]
3505 0000883F 8A35[E6380100] <1> mov dh, [Current_Drv]
3506 00008845 8835[46410100] <1> mov [RUN_CDRV], dh
3507 0000884B 38F2 <1> cmp dl, dh
3508 0000884D 740B <1> je short loc_delfile_change_directory
3509 <1>
3510 0000884F E869E3FFFF <1> call change_current_drive
3511 00008854 0F8225FBFFFF <1> jc loc_file_rw_cmd_failed
3512 <1>
3513 <1> loc_delfile_change_directory:
3514 0000885A 803D[8B420100]20 <1> cmp byte [FindFile_Directory], 20h
3515 00008861 7618 <1> jna short loc_delfile_find
3516 <1>
3517 00008863 FE05[A8FA0000] <1> inc byte [Restore_CDIR]
3518 00008869 BE[8B420100] <1> mov esi, FindFile_Directory
3519 0000886E 30E4 <1> xor ah, ah ; CD_COMMAND sign -> 0
3520 00008870 E8DD120000 <1> call change_current_directory
3521 00008875 0F8204FBFFFF <1> jc loc_file_rw_cmd_failed

```

```

3522 <1>
3523 <1> ;loc_delfile_change_prompt_dir_string:
3524 <1> ;call change_prompt_dir_string
3525 <1>
3526 <1> loc_delfile_find:
3527 <1> ;mov esi, FindFile_Name
3528 0000887B 8B35[48430100] <1> mov esi, [DelFile_FNPointer]
3529 00008881 66B80018 <1> mov ax, 1800h ; Except volume label and dirs
3530 00008885 E8C5F6FFFF <1> call find_first_file
3531 0000888A 0F82EFAFFFFF <1> jc loc_file_rw_cmd_failed
3532 <1>
3533 <1> loc_delfile_ambgfn_check:
3534 00008890 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)
3535 00008893 740B <1> jz short loc_delfile_found
3536 <1>
3537 <1> loc_file_not_found:
3538 00008895 B802000000 <1> mov eax, 2 ; File not found sign
3539 0000889A F9 <1> stc
3540 0000889B E9DFAFFFFF <1> jmp loc_file_rw_cmd_failed
3541 <1>
3542 <1> loc_delfile_found:
3543 000088A0 80E307 <1> and bl, 07h ; Attributes
3544 000088A3 0F85E0FAFFFF <1> jnz loc_permission_denied
3545 <1>
3546 <1> ;loc_delfile_found_save_lnel:
3547 <1> ; mov [DelFile_LNEL], bh ; Long name entry length (if > 0)
3548 <1>
3549 <1> loc_delfile_ask_for_delete:
3550 000088A9 57 <1> push edi ; * (29/02/2016)
3551 <1>
3552 000088AA BE[77FF0000] <1> mov esi, Msg_DoYouWantDelete
3553 000088AF E809DAFFFF <1> call print_msg
3554 000088B4 8B35[48430100] <1> mov esi, [DelFile_FNPointer]
3555 000088BA E8FED9FFFF <1> call print_msg
3556 000088BF BE[2BFF0000] <1> mov esi, Msg_YesNo
3557 000088C4 E8F4D9FFFF <1> call print_msg
3558 <1>
3559 <1> loc_delfile_ask_again:
3560 000088C9 30E4 <1> xor ah, ah
3561 000088CB E84683FFFF <1> call int16h
3562 000088D0 3C1B <1> cmp al, 1Bh
3563 <1> ;je short loc_do_not_delete_file
3564 000088D2 7457 <1> je short loc_delfile_y_n_escape ; 06/03/2016
3565 000088D4 24DF <1> and al, 0DFh
3566 000088D6 A2[35FF0000] <1> mov [Y_N_nextline], al
3567 000088DB 3C59 <1> cmp al, 'Y'
3568 000088DD 7404 <1> je short loc_yes_delete_file
3569 000088DF 3C4E <1> cmp al, 'N'
3570 000088E1 75E6 <1> jne short loc_delfile_ask_again
3571 <1>
3572 <1> loc_do_not_delete_file:
3573 <1> loc_yes_delete_file:
3574 000088E3 A2[35FF0000] <1> mov [Y_N_nextline], al
3575 000088E8 6650 <1> push ax
3576 000088EA BE[35FF0000] <1> mov esi, Y_N_nextline
3577 000088EF E8C9D9FFFF <1> call print_msg
3578 000088F4 6658 <1> pop ax
3579 000088F6 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 000088F7 3C4E <1> cmp al, 'N' ; 'no'
3584 000088F9 0F8480FAFFFF <1> je loc_file_rw_restore_retn
3585 <1>
3586 <1> loc_delete_file:
3587 000088FF 8A3D[8A420100] <1> mov bh, [FindFile_Drv]
3588 <1> ;mov bl, [DelFile_LNEL]
3589 00008905 8A1D[D9420100] <1> mov bl, [FindFile_LongNameEntryLength]
3590 <1> ;mov cx, [DirBuff_EntryCounter]
3591 0000890B 668B0D[04430100] <1> mov cx, [FindFile_DirEntryNumber]
3592 <1> ; (*) EDI = Directory buffer entry offset/address
3593 00008912 E8FD1F0000 <1> call remove_file ; (FILE.ASM, 'proc_delete_file')
3594 00008917 0F8358FAFFFF <1> jnc loc_print_deleted_message
3595 <1>
3596 0000891D 3C05 <1> cmp al, 05h
3597 0000891F 0F8464FAFFFF <1> je loc_permission_denied
3598 00008925 F9 <1> stc
3599 00008926 E954FAFFFF <1> jmp loc_file_rw_cmd_failed
3600 <1>
3601 <1> loc_delfile_y_n_escape:
3602 0000892B B04E <1> mov al, 'N' ; 'no'
3603 0000892D 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_attr')
3609 <1> ; 23/05/2010
3610 <1> ; 17/12/2000 (P2000.ASM)
3611 <1>
3612 <1> ; esi = file or directory name
3613 0000892F 6631C0 <1> xor ax, ax
3614 00008932 66A3[C8FF0000] <1> mov [Attr_Chars], ax
3615 00008938 A2[A0430100] <1> mov [Attributes], al
3616 <1>
3617 <1> get_attrb_fchar:
3618 <1> ; esi = file name
3619 0000893D 8A06 <1> mov al, [esi]
3620 0000893F 3C20 <1> cmp al, 20h
3621 00008941 7623 <1> jna short loc_attr_file_nofilename_retn
3622 <1>
3623 <1> loc_scan_attrb_params:

```

```

3624 00008943 3C2D      <1>      cmp     al, '-'
3625 00008945 0F871C010000      <1>      ja      loc_attr_file_parse_path_name
3626 0000894B 7408      <1>      je      short loc_attr_space
3627                                <1>
3628 0000894D 3C2B      <1>      cmp     al, '+'
3629 0000894F 0F85F8F0FFFF      <1>      jne     loc_cmd_failed
3630                                <1>
3631                                <1> loc_attr_space:
3632 00008955 8A6601      <1>      mov     ah, [esi+1]
3633 00008958 80FC20      <1>      cmp     ah, 20h
3634 0000895B 770A      <1>      ja      short pass_attr_space
3635 0000895D 0F82EAF0FFFF      <1>      jb      loc_cmd_failed
3636 00008963 46      <1>      inc     esi
3637 00008964 EBEF      <1>      jmp     short loc_attr_space
3638                                <1>
3639                                <1> loc_attr_file_nofilename_retn:
3640 00008966 C3      <1>      retn
3641                                <1>
3642                                <1> pass_attr_space:
3643 00008967 80E4DF      <1>      and     ah, 0DFh
3644 0000896A 80FC53      <1>      cmp     ah, 'S'
3645 0000896D 0F87DAF0FFFF      <1>      ja      loc_cmd_failed
3646 00008973 7204      <1>      jb      short pass_attr_system
3647 00008975 B404      <1>      mov     ah, 04h ; System
3648 00008977 EB21      <1>      jmp     short pass_attr_archive
3649                                <1>
3650                                <1> pass_attr_system:
3651 00008979 80FC48      <1>      cmp     ah, 'H'
3652 0000897C 7706      <1>      ja      short pass_attr_hidden
3653 0000897E 7213      <1>      jb      short pass_attr_read_only
3654 00008980 B402      <1>      mov     ah, 02h ; Hidden
3655 00008982 EB16      <1>      jmp     short pass_attr_archive
3656                                <1>
3657                                <1> pass_attr_hidden:
3658 00008984 80FC52      <1>      cmp     ah, 'R'
3659 00008987 0F87C0F0FFFF      <1>      ja      loc_cmd_failed
3660 0000898D 7204      <1>      jb      short pass_attr_read_only ; Read only
3661 0000898F B401      <1>      mov     ah, 01h
3662 00008991 EB07      <1>      jmp     short pass_attr_archive
3663                                <1>
3664                                <1> pass_attr_read_only:
3665 00008993 80FC41      <1>      cmp     ah, 'A'
3666 00008996 753B      <1>      jne     short loc_chk_attr_enter
3667 00008998 B420      <1>      mov     ah, 20h ; Archive
3668                                <1>
3669                                <1> pass_attr_archive:
3670 0000899A 3C2D      <1>      cmp     al, '-'
3671 0000899C 7508      <1>      jne     short pass_reducing_attributes
3672 0000899E 0825[C8FF0000] <1>      or      [Attr_Chars], ah
3673 000089A4 EB06      <1>      jmp     short loc_change_attributes_inc
3674                                <1>
3675                                <1> pass_reducing_attributes:
3676 000089A6 0825[C9FF0000] <1>      or      [Attr_Chars+1], ah
3677                                <1>
3678                                <1> loc_change_attributes_inc:
3679 000089AC 46      <1>      inc     esi
3680 000089AD 8A6601      <1>      mov     ah, [esi+1]
3681 000089B0 80FC20      <1>      cmp     ah, 20h
3682 000089B3 7227      <1>      jb      short pass_change_attr
3683 000089B5 74F5      <1>      je      short loc_change_attributes_inc
3684 000089B7 80FC2D      <1>      cmp     ah, '-'
3685 000089BA 770D      <1>      ja      short loc_chk_next_attr_char1
3686 000089BC 7405      <1>      je      short loc_chk_next_attr_char0
3687 000089BE 80FC2B      <1>      cmp     ah, '+'
3688 000089C1 7506      <1>      jne     short loc_chk_next_attr_char1
3689                                <1>
3690                                <1> loc_chk_next_attr_char0:
3691 000089C3 46      <1>      inc     esi
3692 000089C4 668B06      <1>      mov     ax, [esi]
3693 000089C7 EB9E      <1>      jmp     short pass_attr_space
3694                                <1>
3695                                <1> loc_chk_next_attr_char1:
3696 000089C9 803E2D      <1>      cmp     byte [esi], '-'
3697 000089CC 7799      <1>      ja      short pass_attr_space
3698 000089CE E988000000 <1>      jmp     loc_attr_file_check_fname_fchar
3699                                <1>
3700                                <1> loc_chk_attr_enter:
3701 000089D3 80FC0D      <1>      cmp     ah, 0Dh
3702 000089D6 0F8571F0FFFF      <1>      jne     loc_cmd_failed
3703                                <1>
3704                                <1> pass_change_attr:
3705 000089DC A0[C8FF0000] <1>      mov     al, [Attr_Chars]
3706 000089E1 F6D0      <1>      not     al
3707 000089E3 2005[A0430100] <1>      and     [Attributes], al
3708 000089E9 A0[C9FF0000] <1>      mov     al, [Attr_Chars+1]
3709 000089EE 0805[A0430100] <1>      or      [Attributes], al
3710                                <1>
3711                                <1> loc_show_attributes:
3712 000089F4 BE[13070100] <1>      mov     esi, nextline
3713 000089F9 E8BFD8FFFF <1>      call    print_msg
3714                                <1>
3715                                <1> loc_show_attributes_no_nextline:
3716 000089FE C705[C8FF0000]4E4F- <1>      mov     dword [Attr_Chars], 'NORM'
3716 00008A06 524D      <1>
3717 00008A08 66C705[CCFF0000]41- <1>      mov     word [Attr_Chars+4], 'AL'
3717 00008A10 4C      <1>
3718 00008A11 BE[C8FF0000] <1>      mov     esi, Attr_Chars
3719 00008A16 A0[A0430100] <1>      mov     al, [Attributes]
3720 00008A1B A804      <1>      test    al, 04h
3721 00008A1D 7406      <1>      jz      short pass_put_attr_s
3722 00008A1F 66C7065300 <1>      mov     word [esi], 0053h ; S
3723 00008A24 46      <1>      inc     esi

```



```

3724                                     <1>
3725                                     <1> pass_put_attr_s:
3726 00008A25 A802                       <1>         test    al, 02h
3727 00008A27 7406                       <1>         jz      short pass_put_attr_h
3728 00008A29 66C7064800                 <1>         mov     word [esi], 0048h      ; H
3729 00008A2E 46                         <1>         inc     esi
3730                                     <1>
3731                                     <1> pass_put_attr_h:
3732 00008A2F A801                       <1>         test    al, 01h
3733 00008A31 7406                       <1>         jz      short pass_put_attr_r
3734 00008A33 66C7065200                 <1>         mov     word [esi], 0052h      ; R
3735 00008A38 46                         <1>         inc     esi
3736                                     <1>
3737                                     <1> pass_put_attr_r:
3738 00008A39 3C20                       <1>         cmp     al, 20h
3739 00008A3B 7205                       <1>         jb      short pass_put_attr_a
3740 00008A3D 66C7064100                 <1>         mov     word [esi], 0041h      ; A
3741                                     <1>
3742                                     <1> pass_put_attr_a:
3743 00008A42 BE[BBFF0000]                 <1>         mov     esi, Str_Attributes
3744 00008A47 E871D8FFFF                 <1>         call    print_msg
3745 00008A4C BE[13070100]                 <1>         mov     esi, nextline
3746 00008A51 E867D8FFFF                 <1>         call    print_msg
3747 00008A56 E924F9FFFF                 <1>         jmp     loc_file_rw_restore_retn
3748                                     <1>
3749                                     <1> loc_attr_file_check_fname_fchar:
3750 00008A5B 46                         <1>         inc     esi
3751 00008A5C 803E20                       <1>         cmp     byte [esi], 20h
3752 00008A5F 74FA                       <1>         je      short loc_attr_file_check_fname_fchar
3753 00008A61 0F8275FFFFFF                 <1>         jnb     pass_change_attr
3754                                     <1>
3755                                     <1> loc_attr_file_parse_path_name:
3756 00008A67 BF[8A420100]                 <1>         mov     edi, FindFile_Drv
3757 00008A6C E8F5160000                 <1>         call    parse_path_name
3758 00008A71 0F82D6EFFFFFFF                 <1>         jc      loc_cmd_failed
3759                                     <1>
3760                                     <1> loc_attr_file_check_filename_exists:
3761 00008A77 BE[CC420100]                 <1>         mov     esi, FindFile_Name
3762 00008A7C 803E20                       <1>         cmp     byte [esi], 20h
3763 00008A7F 0F86C8EFFFFFFF                 <1>         jna     loc_cmd_failed
3764 00008A85 8935[48430100]                 <1>         mov     [DelFile_FNPointer], esi
3765                                     <1>
3766                                     <1> loc_attr_file_drv:
3767 00008A8B 8A35[E6380100]                 <1>         mov     dh, [Current_Drv]
3768 00008A91 8835[46410100]                 <1>         mov     [RUN_CDRV], dh
3769                                     <1>
3770 00008A97 8A15[8A420100]                 <1>         mov     dl, [FindFile_Drv]
3771 00008A9D 38F2                       <1>         cmp     dl, dh
3772 00008A9F 740B                       <1>         je      short loc_attr_file_change_directory
3773                                     <1>
3774 00008AA1 E817E1FFFF                 <1>         call    change_current_drive
3775 00008AA6 0F82D3F8FFFFFF                 <1>         jc      loc_file_rw_cmd_failed
3776                                     <1>
3777                                     <1> loc_attr_file_change_directory:
3778 00008AAC 803D[8B420100]20                 <1>         cmp     byte [FindFile_Directory], 20h
3779 00008AB3 7618                       <1>         jna     short loc_attr_file_find
3780                                     <1>
3781 00008AB5 FE05[A8FA0000]                 <1>         inc     byte [Restore_CDIR]
3782                                     <1>
3783 00008ABB BE[8B420100]                 <1>         mov     esi, FindFile_Directory
3784 00008AC0 30E4                       <1>         xor     ah, ah ; CD_COMMAND sign -> 0
3785 00008AC2 E88B100000                 <1>         call    change_current_directory
3786 00008AC7 0F82B2F8FFFFFF                 <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 00008ACD 8B35[48430100]                 <1>         mov     esi, [DelFile_FNPointer]
3794 00008AD3 66B80008                 <1>         mov     ax, 0800h ; Except volume labels
3795 00008AD7 E873F4FFFF                 <1>         call    find_first_file
3796 00008ADC 0F829DF8FFFFFF                 <1>         jc      loc_file_rw_cmd_failed
3797                                     <1>
3798                                     <1> loc_attr_file_ambgfn_check:
3799 00008AE2 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 00008AE5 0F85AAFDFFFFFFFF                 <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 00008AEB 66833D[C8FF0000]00                 <1>         cmp     word [Attr_Chars], 0
3815 00008AF3 770B                       <1>         ja      short loc_attr_file_change_attributes
3816 00008AF5 881D[A0430100]                 <1>         mov     [Attributes], bl
3817 00008AFB E9F4FEFFFFFF                 <1>         jmp     loc_show_attributes
3818                                     <1>
3819                                     <1> loc_attr_file_change_attributes:
3820 00008B00 A0[C8FF0000]                 <1>         mov     al, [Attr_Chars]
3821 00008B05 F6D0                       <1>         not     al
3822 00008B07 20C3                       <1>         and     bl, al
3823 00008B09 A0[C9FF0000]                 <1>         mov     al, [Attr_Chars+1]
3824 00008B0E 08C3                       <1>         or      bl, al
3825                                     <1>

```

```

3826 00008B10 66817F0CA101 <1>      cmp     word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
3827 00008B16 741D <1>      je      short loc_attr_file_fs_check
3828 <1>
3829 00008B18 881D[A0430100] <1>      mov     [Attributes], bl
3830 00008B1E 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 00008B21 C605[10410100]02 <1>      mov     byte [DirBuff_ValidData], 2
3838 <1>
3839 00008B28 E89D1A0000 <1>      call    save_directory_buffer
3840 00008B2D 0F824CF8FFFF <1>      jc      loc_file_rw_cmd_failed
3841 <1>
3842 00008B33 EB33 <1>      jmp     short loc_print_attr_changed_message
3843 <1>
3844 <1> loc_attr_file_fs_check:
3845 00008B35 29C0 <1>      sub     eax, eax
3846 00008B37 8A25[0E410100] <1>      mov     ah, [DirBuff_DRV]
3847 00008B3D BE00010900 <1>      mov     esi, Logical_DOSDisks
3848 00008B42 01C6 <1>      add     esi, eax
3849 00008B44 807E04A1 <1>      cmp     byte [esi+LD_FSType], 0A1h
3850 00008B48 7309 <1>      jnc     short loc_attr_file_change_fs_file_attributes
3851 00008B4A 66B80D00 <1>      mov     ax, 0Dh ; Invalid Data
3852 00008B4E 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 00008B53 88D8 <1>      mov     al, bl ; File/Directory Attributes
3857 00008B55 30E4 <1>      xor     ah, ah ; Attributes in MS-DOS format sign
3858 00008B57 E8A7050000 <1>      call    change_fs_file_attributes
3859 00008B5C 0F821DF8FFFF <1>      jc      loc_file_rw_cmd_failed
3860 <1>
3861 00008B62 881D[A0430100] <1>      mov     [Attributes], bl
3862 <1>
3863 <1> loc_print_attr_changed_message:
3864 00008B68 BE[B6FF0000] <1>      mov     esi, Msg_New
3865 00008B6D E84BD7FFFF <1>      call    print_msg
3866 00008B72 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 00008B77 803E20 <1>      cmp     byte [esi], 20h
3880 00008B7A 7614 <1>      jna     short loc_rename_nofilename_retn
3881 <1>
3882 00008B7C 8935[C8430100] <1>      mov     [SourceFilePath], esi
3883 <1>
3884 <1> rename_scan_source_file:
3885 00008B82 46 <1>      inc     esi
3886 00008B83 803E20 <1>      cmp     byte [esi], 20h
3887 00008B86 7409 <1>      je      short rename_scan_destination_file_1
3888 <1>      ;jb     short loc_rename_nofilename_retn
3889 00008B88 0F82BFEEFFFF <1>      jb      loc_cmd_failed
3890 00008B8E EBF2 <1>      jmp     short rename_scan_source_file
3891 <1>
3892 <1> loc_rename_nofilename_retn: ; 08/03/2016
3893 00008B90 C3 <1>      retn
3894 <1>
3895 <1> rename_scan_destination_file_1:
3896 00008B91 C60600 <1>      mov     byte [esi], 0
3897 <1>
3898 <1> rename_scan_destination_file_2:
3899 00008B94 46 <1>      inc     esi
3900 00008B95 803E20 <1>      cmp     byte [esi], 20h
3901 00008B98 74FA <1>      je      short rename_scan_destination_file_2
3902 <1>      ;jb     short loc_rename_nofilename_retn
3903 00008B9A 0F82ADEEFFFF <1>      jb      loc_cmd_failed
3904 <1>
3905 00008BA0 8935[CC430100] <1>      mov     [DestinationFilePath], esi
3906 <1>
3907 <1> rename_scan_destination_file_3:
3908 00008BA6 46 <1>      inc     esi
3909 00008BA7 803E20 <1>      cmp     byte [esi], 20h
3910 00008BAA 77FA <1>      ja      short rename_scan_destination_file_3
3911 <1>
3912 00008BAC C60600 <1>      mov     byte [esi], 0
3913 <1>
3914 <1> loc_rename_save_current_drive:
3915 00008BAF 8A35[E6380100] <1>      mov     dh, [Current_Drv]
3916 00008BB5 8835[46410100] <1>      mov     byte [RUN_CDRV], dh
3917 <1>
3918 <1> loc_rename_sf_parse_path_name:
3919 00008BBB 8B35[C8430100] <1>      mov     esi, [SourceFilePath]
3920 00008BC1 BF[8A420100] <1>      mov     edi, FindFile_Drv
3921 00008BC6 E89B150000 <1>      call    parse_path_name
3922 00008BCB 0F827CEEFFFF <1>      jc      loc_cmd_failed
3923 <1>
3924 <1> loc_rename_sf_check_filename_exists:
3925 00008BD1 BE[CC420100] <1>      mov     esi, FindFile_Name
3926 00008BD6 803E20 <1>      cmp     byte [esi], 20h
3927 00008BD9 0F866EEFFFFF <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 00008BDF 8A15[8A420100]             <1>         mov     dl, [FindFile_Drv]
3936 00008BE5 38F2                       <1>         cmp     dl, dh ; dh = [Current_Drv]
3937 00008BE7 740B                       <1>         je      short rename_sf_change_directory
3938                                     <1>
3939 00008BE9 E8CFDFFFFFFF               <1>         call    change_current_drive
3940 00008BEE 0F828BF7FFFF               <1>         jc      loc_file_rw_cmd_failed
3941                                     <1>
3942                                     <1> rename_sf_change_directory:
3943 00008BF4 803D[8B420100]20           <1>         cmp     byte [FindFile_Directory], 20h
3944 00008BFB 7618                       <1>         jna     short rename_sf_find
3945                                     <1>
3946 00008BFD FE05[A8FA0000]             <1>         inc     byte [Restore_CDIR]
3947 00008C03 BE[8B420100]               <1>         mov     esi, FindFile_Directory
3948 00008C08 30E4                       <1>         xor     ah, ah ; CD_COMMAND sign -> 0
3949 00008C0A E8430F0000                 <1>         call    change_current_directory
3950 00008C0F 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 00008C15 BE[CC420100]               <1>         mov     esi, FindFile_Name
3958                                     <1>
3959 00008C1A 66B80008                   <1>         mov     ax, 0800h ; Except volume labels
3960 00008C1E E82CF3FFFF               <1>         call    find_first_file
3961 00008C23 0F8256F7FFFF               <1>         jc      loc_file_rw_cmd_failed
3962                                     <1>
3963                                     <1> loc_rename_sf_ambgfn_check:
3964 00008C29 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 00008C2C 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 00008C32 F6C307                     <1>         test    bl, 07h ; Attributes, S-H-R
3980 00008C35 0F854EF7FFFF               <1>         jnz     loc_permission_denied
3981                                     <1>
3982 00008C3B BE[8A420100]               <1>         mov     esi, FindFile_Drv
3983 00008C40 BF[D0430100]               <1>         mov     edi, SourceFile_Drv
3984 00008C45 B920000000                 <1>         mov     ecx, 32
3985 00008C4A F3A5                       <1>         rep     movsd
3986                                     <1>
3987                                     <1> loc_rename_df_parse_path_name:
3988 00008C4C 8B35[CC430100]             <1>         mov     esi, [DestinationFilePath]
3989 00008C52 BF[8A420100]               <1>         mov     edi, FindFile_Drv
3990 00008C57 E80A150000                 <1>         call    parse_path_name
3991 00008C5C 7219                       <1>         jc      short loc_rename_df_cmd_failed
3992                                     <1>
3993                                     <1>         ;mov     dh, [RUN_CDRV]
3994 00008C5E 8A35[E6380100]             <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 00008C64 8A15[8A420100]             <1>         mov     dl, [FindFile_Drv]
4000 00008C6A 38F2                       <1>         cmp     dl, dh ; are source and destination drives different ?!
4001 00008C6C 7509                       <1>         jne     short loc_rename_df_cmd_failed ; yes!
4002                                     <1>
4003                                     <1> rename_df_check_dirname_exists:
4004 00008C6E 803D[8B420100]00           <1>         cmp     byte [FindFile_Directory], 0
4005 00008C75 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 00008C77 B801000000                 <1>         mov     eax, 1 ; TRDOS 'Bad command or file name' error
4010 00008C7C F9                         <1>         stc
4011 00008C7D E9FDF6FFFF               <1>         jmp     loc_file_rw_cmd_failed
4012                                     <1>
4013                                     <1> rename_df_check_filename_exists:
4014 00008C82 BE[CC420100]               <1>         mov     esi, FindFile_Name
4015 00008C87 E883F6FFFF               <1>         call    check_filename
4016 00008C8C 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 00008C92 0FB6F6                     <1>         movzx   esi, dh
4026                                     <1>         ;movzx   esi, byte [Current_Drv]
4027 00008C95 81C600010900               <1>         add     esi, Logical_DOSDisks
4028                                     <1>
4029 00008C9B 88F2                       <1>         mov     dl, dh ; dl = [Current_Drv]

```

```

4030 00008C9D 8A7601      <1>      mov     dh, [esi+LD_DiskType]
4031                                <1>
4032 00008CA0 80FE01      <1>      cmp     dh, 1 ; 0 = Invalid
4033 00008CA3 7310       <1>      jnb     short rename_df_compare_sf_df_name
4034                                <1>
4035                                <1>      ; 16/10/2016 (13h -> 30)
4036 00008CA5 B81E000000      <1>      mov     eax, 30 ; 'Disk write-protected' error
4037 00008CAA 8B1D[CC430100] <1>      mov     ebx, [DestinationFilePath]
4038 00008CB0 E9CAF6FFFF      <1>      jmp     loc_file_rw_cmd_failed
4039                                <1>
4040                                <1> rename_df_compare_sf_df_name:
4041 00008CB5 BE[CC420100] <1>      mov     esi, FindFile_Name
4042 00008CBA BF[12440100] <1>      mov     edi, SourceFile_Name
4043 00008CBF B90C000000      <1>      mov     ecx, 12
4044                                <1> rename_df_compare_sf_df_name_next:
4045 00008CC4 AC          <1>      lodsb
4046 00008CC5 AE          <1>      scasb
4047 00008CC6 7506       <1>      jne     short loc_rename_df_find
4048 00008CC8 08C0       <1>      or      al, al
4049 00008CCA 74AB       <1>      jz      short loc_rename_df_cmd_failed
4050 00008CCC 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 00008CCE BE[CC420100] <1>      mov     esi, FindFile_Name
4055                                <1>
4056 00008CD3 6631C0      <1>      xor     ax, ax ; Any
4057 00008CD6 E874F2FFFF      <1>      call    find_first_file
4058 00008CDB 730A       <1>      jnc     short loc_rename_df_found
4059                                <1>
4060                                <1> loc_rename_df_check_error_code:
4061                                <1>      ;cmp     eax, 2
4062 00008CDD 3C02       <1>      cmp     al, 2 ; Not found error
4063 00008CDF 7411       <1>      je      short rename_df_move_find_struct_to_dest
4064 00008CE1 F9          <1>      stc
4065 00008CE2 E998F6FFFF      <1>      jmp     loc_file_rw_cmd_failed
4066                                <1>
4067                                <1> loc_rename_df_found:
4068                                <1>      ; 05/11/2016
4069 00008CE7 B805000000      <1>      mov     eax, 05h ; permission denied error
4070 00008CEC F9          <1>      stc
4071 00008CED E997F6FFFF      <1>      jmp     loc_permission_denied ; 06/11/2016
4072                                <1>
4073                                <1> rename_df_move_find_struct_to_dest:
4074 00008CF2 BE[8A420100] <1>      mov     esi, FindFile_Drv
4075 00008CF7 BF[50440100] <1>      mov     edi, DestinationFile_Drv
4076 00008CFC B920000000      <1>      mov     ecx, 32
4077 00008D01 F3A5       <1>      rep     movsd
4078                                <1>
4079                                <1> loc_rename_df_process_q_sf:
4080                                <1>      ;mov     ecx, 12
4081 00008D03 B10C       <1>      mov     cl, 12
4082 00008D05 BE[12440100] <1>      mov     esi, SourceFile_Name
4083 00008D0A BF[F7FF0000] <1>      mov     edi, Rename_OldName
4084                                <1> rename_df_process_q_nml_1_sf:
4085 00008D0F AC          <1>      lodsb
4086 00008D10 3C20       <1>      cmp     al, 20h
4087 00008D12 7603       <1>      jna     short rename_df_process_q_nml_2_sf
4088 00008D14 AA          <1>      stosb
4089 00008D15 E2F8       <1>      loop    rename_df_process_q_nml_1_sf
4090                                <1>
4091                                <1> rename_df_process_q_nml_2_sf:
4092 00008D17 C60700      <1>      mov     byte [edi], 0
4093                                <1>
4094                                <1> loc_rename_df_process_q_df:
4095                                <1>      ;mov     ecx, 12
4096 00008D1A B10C       <1>      mov     cl, 12
4097 00008D1C BE[92440100] <1>      mov     esi, DestinationFile_Name
4098 00008D21 BF[08000100] <1>      mov     edi, Rename_NewName
4099                                <1> rename_df_process_q_nml_1_df:
4100 00008D26 AC          <1>      lodsb
4101 00008D27 3C20       <1>      cmp     al, 20h
4102 00008D29 7603       <1>      jna     short loc_rename_df_process_q_nml_2_df
4103 00008D2B AA          <1>      stosb
4104 00008D2C E2F8       <1>      loop    rename_df_process_q_nml_1_df
4105                                <1>
4106                                <1> loc_rename_df_process_q_nml_2_df:
4107 00008D2E C60700      <1>      mov     byte [edi], 0
4108                                <1>
4109                                <1> loc_rename_confirmation_question:
4110 00008D31 BE[CFFF0000] <1>      mov     esi, Msg_DoYouWantRename
4111 00008D36 E882D5FFFF      <1>      call    print_msg
4112                                <1>
4113 00008D3B A0[2D440100] <1>      mov     al, [SourceFile_DirEntry+11] ; Attributes
4114 00008D40 2410       <1>      and     al, 10h
4115 00008D42 750C       <1>      jnz     short rename_confirmation_question_dir
4116                                <1>
4117                                <1> rename_confirmation_question_file:
4118 00008D44 BE[E6FF0000] <1>      mov     esi, Rename_File
4119 00008D49 E86FD5FFFF      <1>      call    print_msg
4120 00008D4E EB0A       <1>      jmp     short rename_confirmation_question_as
4121                                <1>
4122                                <1> rename_confirmation_question_dir:
4123 00008D50 BE[ECFF0000] <1>      mov     esi, Rename_Directory
4124 00008D55 E863D5FFFF      <1>      call    print_msg
4125                                <1>
4126                                <1> rename_confirmation_question_as:
4127 00008D5A BE[F7FF0000] <1>      mov     esi, Rename_OldName
4128 00008D5F E859D5FFFF      <1>      call    print_msg
4129 00008D64 BE[04000100] <1>      mov     esi, Msg_File_rename_as
4130 00008D69 E84FD5FFFF      <1>      call    print_msg
4131 00008D6E BE[2BFF0000] <1>      mov     esi, Msg_YesNo

```



```

4132 00008D73 E845D5FFFF <1>      call  print_msg
4133 <1>
4134 <1> loc_rename_ask_again:
4135 00008D78 30E4 <1>      xor    ah, ah
4136 00008D7A E8977EFFFF <1>      call  int16h
4137 00008D7F 3C1B <1>      cmp    al, 1Bh
4138 00008D81 740F <1>      je     short loc_do_not_rename_file
4139 00008D83 24DF <1>      and    al, 0DFh
4140 00008D85 A2[35FF0000] <1>      mov    [Y_N_nextline], al
4141 00008D8A 3C59 <1>      cmp    al, 'Y'
4142 00008D8C 7404 <1>      je     short loc_yes_rename_file
4143 00008D8E 3C4E <1>      cmp    al, 'N'
4144 00008D90 75E6 <1>      jne    short loc_rename_ask_again
4145 <1>
4146 <1> loc_do_not_rename_file:
4147 <1> loc_yes_rename_file:
4148 00008D92 A2[35FF0000] <1>      mov    [Y_N_nextline], al
4149 00008D97 6650 <1>      push   ax
4150 00008D99 BE[35FF0000] <1>      mov    esi, Y_N_nextline
4151 00008D9E E81AD5FFFF <1>      call  print_msg
4152 00008DA3 6658 <1>      pop    ax
4153 <1>      ;cmp    al, 'Y' ; 'yes'
4154 <1>      ;cmc
4155 <1>      ;jnc loc_file_rw_restore_retn
4156 00008DA5 3C4E <1>      cmp    al, 'N' ; 'no'
4157 00008DA7 0F84D2F5FFFF <1>      je     loc_file_rw_restore_retn
4158 <1>
4159 00008DAD BE[08000100] <1>      mov    esi, Rename_NewName
4160 00008DB2 668B0D[4A440100] <1>      mov    cx, [SourceFile_DirEntryNumber]
4161 00008DB9 66A1[36440100] <1>      mov    ax, [SourceFile_DirEntry+20] ; First Cluster, HW
4162 00008DBF 66C1E010 <1>      shl    ax, 16
4163 00008DC3 66A1[3C440100] <1>      mov    ax, [SourceFile_DirEntry+26] ; First Cluster, LW
4164 <1>
4165 00008DC9 0FB61D[1F440100] <1>      movzx   ebx, byte [SourceFile_LongNameEntryLength]
4166 00008DD0 E8DB1B0000 <1>      call  rename_directory_entry
4167 00008DD5 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 00008DDA 803E20 <1>      cmp    byte [esi], 20h
4184 00008DDD 7614 <1>      jna    short loc_move_nofilename_retn
4185 <1>
4186 00008DDF 8935[C8430100] <1>      mov    [SourceFilePath], esi
4187 <1>
4188 <1> move_scan_source_file:
4189 00008DE5 46 <1>      inc    esi
4190 00008DE6 803E20 <1>      cmp    byte [esi], 20h
4191 00008DE9 7409 <1>      je     short move_scan_destination_1
4192 <1>      ;jb    short loc_move_nofilename_retn
4193 00008DEB 0F825CECFFFF <1>      jb     loc_cmd_failed
4194 00008DF1 EBF2 <1>      jmp    short move_scan_source_file
4195 <1>
4196 <1> loc_move_nofilename_retn:
4197 00008DF3 C3 <1>      retn
4198 <1>
4199 <1> move_scan_destination_1:
4200 00008DF4 C60600 <1>      mov    byte [esi], 0
4201 <1>
4202 <1> move_scan_destination_2:
4203 00008DF7 46 <1>      inc    esi
4204 00008DF8 803E20 <1>      cmp    byte [esi], 20h
4205 00008DFB 74FA <1>      je     short move_scan_destination_2
4206 <1>      ;jb    short loc_move_nofilename_retn
4207 00008DFD 0F824AECFFFF <1>      jb     loc_cmd_failed
4208 <1>
4209 00008E03 8935[CC430100] <1>      mov    [DestinationFilePath], esi
4210 <1>
4211 <1> move_scan_destination_3:
4212 00008E09 46 <1>      inc    esi
4213 00008E0A 803E20 <1>      cmp    byte [esi], 20h
4214 00008E0D 77FA <1>      ja     short move_scan_destination_3
4215 00008E0F C60600 <1>      mov    byte [esi], 0
4216 <1>
4217 <1> loc_move_scan_destination_OK:
4218 00008E12 8B35[C8430100] <1>      mov    esi, [SourceFilePath]
4219 00008E18 8B3D[CC430100] <1>      mov    edi, [DestinationFilePath]
4220 <1>
4221 00008E1E B001 <1>      mov    al, 1 ; move procedure Phase 1
4222 00008E20 E8081C0000 <1>      call  move_source_file_to_destination_file
4223 00008E25 7328 <1>      jnc    short move_source_file_to_destination_question
4224 <1>
4225 <1> loc_move_cmd_failed_1:
4226 00008E27 08C0 <1>      or     al, al
4227 00008E29 0F841EECFFFF <1>      jz     loc_cmd_failed
4228 00008E2F 3C11 <1>      cmp    al, 11h
4229 00008E31 740D <1>      je     short loc_msg_not_same_device
4230 00008E33 3C05 <1>      cmp    al, 05h
4231 00008E35 0F853DECFFFF <1>      jne    loc_run_cmd_failed
4232 <1>
4233 00008E3B 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 00008E40 BE[15000100] <1>      mov  esi, msg_not_same_drv
4241 00008E45 E873D4FFFF <1>      call print_msg
4242 00008E4A E930F5FFFF <1>      jmp  loc_file_rw_restore_retn
4243 <1>
4244 <1> move_source_file_to_destination_question:
4245 00008E4F A0[D0430100] <1>      mov  al, [SourceFile_Drv]
4246 00008E54 0441 <1>      add  al, 'A'
4247 00008E56 A2[77000100] <1>      mov  [msg_source_file_drv], al
4248 00008E5B A0[50440100] <1>      mov  al, [DestinationFile_Drv]
4249 00008E60 0441 <1>      add  al, 'A'
4250 00008E62 A2[96000100] <1>      mov  [msg_destination_file_drv], al
4251 <1>
4252 00008E67 57 <1>      push edi ; *
4253 <1>
4254 00008E68 BE[5B000100] <1>      mov  esi, msg_source_file
4255 00008E6D E84BD4FFFF <1>      call print_msg
4256 00008E72 BE[D1430100] <1>      mov  esi, SourceFile_Directory
4257 00008E77 803E20 <1>      cmp  byte [esi], 20h
4258 00008E7A 7605 <1>      jna  short msftdfq_sfn
4259 00008E7C E83CD4FFFF <1>      call print_msg
4260 <1> msftdfq_sfn:
4261 00008E81 BE[12440100] <1>      mov  esi, SourceFile_Name
4262 00008E86 E832D4FFFF <1>      call print_msg
4263 00008E8B BE[7A000100] <1>      mov  esi, msg_destination_file
4264 00008E90 E828D4FFFF <1>      call print_msg
4265 00008E95 BE[51440100] <1>      mov  esi, DestinationFile_Directory
4266 00008E9A 803E20 <1>      cmp  byte [esi], 20h
4267 00008E9D 7605 <1>      jna  short msftdfq_dfn
4268 00008E9F E819D4FFFF <1>      call print_msg
4269 <1> msftdfq_dfn:
4270 00008EA4 BE[92440100] <1>      mov  esi, DestinationFile_Name
4271 00008EA9 E80FD4FFFF <1>      call print_msg
4272 00008EAE BE[99000100] <1>      mov  esi, msg_copy_nextline
4273 00008EB3 E805D4FFFF <1>      call print_msg
4274 00008EB8 BE[99000100] <1>      mov  esi, msg_copy_nextline
4275 00008EBD E8FBD3FFFF <1>      call print_msg
4276 <1>
4277 <1> loc_move_ask_for_new_file_yes_no:
4278 00008EC2 BE[27000100] <1>      mov  esi, Msg_DoYouWantMoveFile
4279 00008EC7 E8F1D3FFFF <1>      call print_msg
4280 00008ECC BE[2BFF0000] <1>      mov  esi, Msg_YesNo
4281 00008ED1 E8E7D3FFFF <1>      call print_msg
4282 <1> loc_move_ask_for_new_file_again:
4283 00008ED6 30E4 <1>      xor  ah, ah
4284 00008ED8 E8397DFFFF <1>      call int16h
4285 00008EDD 3C1B <1>      cmp  al, 1Bh
4286 <1>      ;je  short loc_do_not_move_file
4287 00008EDF 744F <1>      je  short loc_move_y_n_escape
4288 00008EE1 24DF <1>      and  al, 0DFh
4289 00008EE3 A2[35FF0000] <1>      mov  [Y_N_nextline], al
4290 00008EE8 3C59 <1>      cmp  al, 'Y'
4291 00008EEA 7404 <1>      je  short loc_yes_move_file
4292 00008EEC 3C4E <1>      cmp  al, 'N'
4293 00008EEE 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 00008EF0 A2[35FF0000] <1>      mov  [Y_N_nextline], al
4298 00008EF5 6650 <1>      push ax
4299 00008EF7 BE[35FF0000] <1>      mov  esi, Y_N_nextline
4300 00008EFC E8BCD3FFFF <1>      call print_msg
4301 00008F01 6658 <1>      pop  ax
4302 00008F03 5F <1>      pop  edi ; *
4303 <1>      ;cmp  al, 'Y' ; 'yes'
4304 <1>      ;cmc
4305 <1>      ;jnc loc_file_rw_restore_retn
4306 00008F04 3C4E <1>      cmp  al, 'N' ; 'no'
4307 00008F06 0F8473F4FFFF <1>      je  loc_file_rw_restore_retn
4308 <1>
4309 <1> loc_move_yes_move_file:
4310 00008F0C B002 <1>      mov  al, 2 ; move procedure Phase 2
4311 00008F0E E81A1B0000 <1>      call move_source_file_to_destination_file
4312 <1>      ;jc  short loc_move_cmd_failed_2
4313 00008F13 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 00008F19 3C27 <1>      cmp  al, 27h
4322 00008F1B 0F8557EBFFFF <1>      jne  loc_run_cmd_failed
4323 <1>
4324 00008F21 BE[40000100] <1>      mov  esi, msg_insufficient_disk_space
4325 00008F26 E892D3FFFF <1>      call print_msg
4326 <1>
4327 00008F2B E94FF4FFFF <1>      jmp  loc_file_rw_restore_retn
4328 <1>
4329 <1> loc_move_y_n_escape:
4330 00008F30 B04E <1>      mov  al, 'N' ; 'no'
4331 00008F32 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      <1>      cmp     byte [esi], 20h
4344      <1>      jna     short loc_copy_nofilename_retn
4345      <1>
4346      <1>      mov     [SourceFilePath], esi
4347      <1>
4348      <1> copy_scan_source_file:
4349      <1>      inc     esi
4350      <1>      cmp     byte [esi], 20h
4351      <1>      je      short copy_scan_destination_1
4352      <1>      ;jb     short loc_copy_nofilename_retn
4353      <1>      jb      loc_cmd_failed
4354      <1>      jmp     short copy_scan_source_file
4355      <1>
4356      <1> loc_copy_nofilename_retn:
4357      <1>      retn
4358      <1>
4359      <1> copy_scan_destination_1:
4360      <1>      mov     byte [esi], 0
4361      <1>
4362      <1> copy_scan_destination_2:
4363      <1>      inc     esi
4364      <1>      cmp     byte [esi], 20h
4365      <1>      je      short copy_scan_destination_2
4366      <1>      ;jb     short loc_copy_nofilename_retn
4367      <1>      jb      loc_cmd_failed
4368      <1>
4369      <1>      mov     [DestinationFilePath], esi
4370      <1>
4371      <1> copy_scan_destination_3:
4372      <1>      inc     esi
4373      <1>      cmp     byte [esi], 20h
4374      <1>      ja      short copy_scan_destination_3
4375      <1>      mov     byte [esi], 0
4376      <1>
4377      <1> loc_copy_save_current_drive:
4378      <1>      mov     dh, [Current_Drv]
4379      <1>      mov     [RUN_CDRV], dh
4380      <1>
4381      <1> copy_source_file_to_destination_phase_1:
4382      <1>      mov     esi, [SourceFilePath]
4383      <1>      mov     edi, [DestinationFilePath]
4384      <1>
4385      <1>      mov     al, 1 ; copy procedure Phase 1
4386      <1>      call    copy_source_file_to_destination_file
4387      <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      <1>      or      al, al
4392      <1>      jnz     short loc_copy_cmd_failed_2
4393      <1>
4394      <1>      inc     al ; mov al, 1 ; Bad command or file name !
4395      <1>      jmp     loc_run_cmd_failed
4396      <1>
4397      <1> loc_copy_cmd_failed_2:
4398      <1>      cmp     al, 27h ; Insufficient disk space
4399      <1>      je      short loc_file_write_insuff_disk_space_msg
4400      <1>
4401      <1>      cmp     al, 05h
4402      <1>      jne     loc_run_cmd_failed
4403      <1>
4404      <1>      jmp     loc_permission_denied
4405      <1>
4406      <1> loc_file_write_insuff_disk_space_msg:
4407      <1>      mov     esi, msg_insufficient_disk_space
4408      <1>      call    print_msg
4409      <1>      jmp     loc_file_rw_restore_retn
4410      <1>
4411      <1> copy_source_file_to_destination_question:
4412      <1>      push     edi ; *
4413      <1>
4414      <1>      ; dh = source file attributes
4415      <1>      ; dl > 0 -> destination file found
4416      <1>      and     dl, dl
4417      <1>      jz      short copy_source_file_to_destination_pass_owrq
4418      <1>
4419      <1> loc_copy_ask_for_owr_yes_no:
4420      <1>      mov     esi, Msg_DoYouWantOverWriteFile
4421      <1>      call    print_msg
4422      <1>      mov     esi, DestinationFile_Name
4423      <1>      call    print_msg
4424      <1>      mov     esi, Msg_YesNo
4425      <1>      call    print_msg
4426      <1>
4427      <1> loc_copy_ask_for_owr_again:
4428      <1>      xor     ah, ah
4429      <1>      call    int16h
4430      <1>      cmp     al, 1Bh
4431      <1>      ;je     loc_do_not_copy_file
4432      <1>      je      short loc_copy_y_n_escape
4433      <1>      and     al, 0DFh
4434      <1>      mov     [Y_N_nextline], al
4435      <1>      cmp     al, 'Y'
4436      <1>      je      loc_yes_copy_file
4437      <1>      cmp     al, 'N'

```

```

4438 00008FF7 0F84A9000000 <1> je loc_do_not_copy_file
4439 00008FFD EBDC <1> jmp short loc_copy_ask_for_owr_again
4440 <1>
4441 <1> loc_copy_y_n_escape:
4442 00008FFF B04E <1> mov al, 'N' ; 'no'
4443 00009001 E9A0000000 <1> jmp loc_do_not_copy_file
4444 <1>
4445 <1> copy_source_file_to_destination_pass_owrq:
4446 00009006 A0[D0430100] <1> mov al, [SourceFile_Drv]
4447 0000900B 0441 <1> add al, 'A'
4448 0000900D A2[77000100] <1> mov [msg_source_file_drv], al
4449 00009012 A0[50440100] <1> mov al, [DestinationFile_Drv]
4450 00009017 0441 <1> add al, 'A'
4451 00009019 A2[96000100] <1> mov [msg_destination_file_drv], al
4452 <1>
4453 0000901E BE[5B000100] <1> mov esi, msg_source_file
4454 00009023 E895D2FFFF <1> call print_msg
4455 00009028 BE[D1430100] <1> mov esi, SourceFile_Directory
4456 0000902D 803E20 <1> cmp byte [esi], 20h
4457 00009030 7605 <1> jna short csftdfq_sfn
4458 00009032 E886D2FFFF <1> call print_msg
4459 <1> csftdfq_sfn:
4460 00009037 BE[12440100] <1> mov esi, SourceFile_Name
4461 0000903C E87CD2FFFF <1> call print_msg
4462 00009041 BE[7A000100] <1> mov esi, msg_destination_file
4463 00009046 E872D2FFFF <1> call print_msg
4464 0000904B BE[51440100] <1> mov esi, DestinationFile_Directory
4465 00009050 803E20 <1> cmp byte [esi], 20h
4466 00009053 7605 <1> jna short csftdfq_dfn
4467 00009055 E863D2FFFF <1> call print_msg
4468 <1> csftdfq_dfn:
4469 0000905A BE[92440100] <1> mov esi, DestinationFile_Name
4470 0000905F E859D2FFFF <1> call print_msg
4471 00009064 BE[99000100] <1> mov esi, msg_copy_nextline
4472 00009069 E84FD2FFFF <1> call print_msg
4473 0000906E BE[99000100] <1> mov esi, msg_copy_nextline
4474 00009073 E845D2FFFF <1> call print_msg
4475 <1>
4476 <1> loc_copy_ask_for_new_file_yes_no:
4477 00009078 BE[BB000100] <1> mov esi, Msg_DoYouWantCopyFile
4478 0000907D E83BD2FFFF <1> call print_msg
4479 00009082 BE[2BFF0000] <1> mov esi, Msg_YesNo
4480 00009087 E831D2FFFF <1> call print_msg
4481 <1>
4482 <1> loc_copy_ask_for_new_file_again:
4483 0000908C 30E4 <1> xor ah, ah
4484 0000908E E8837BFFFF <1> call int16h
4485 00009093 3C1B <1> cmp al, 1Bh
4486 00009095 740F <1> je short loc_do_not_copy_file
4487 00009097 24DF <1> and al, 0DFh
4488 00009099 A2[35FF0000] <1> mov [Y_N_nextline], al
4489 0000909E 3C59 <1> cmp al, 'Y'
4490 000090A0 7404 <1> je short loc_yes_copy_file
4491 000090A2 3C4E <1> cmp al, 'N'
4492 000090A4 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 000090A6 A2[35FF0000] <1> mov [Y_N_nextline], al
4497 000090AB 6650 <1> push ax
4498 000090AD BE[35FF0000] <1> mov esi, Y_N_nextline
4499 000090B2 E806D2FFFF <1> call print_msg
4500 000090B7 6658 <1> pop ax
4501 000090B9 5F <1> pop edi ; *
4502 <1> ;cmp al, 'Y' ; 'yes'
4503 <1> ;cmc
4504 <1> ;jnc loc_file_rw_restore_retn
4505 000090BA 3C4E <1> cmp al, 'N' ; 'no'
4506 000090BC 0F84BDF2FFFF <1> je loc_file_rw_restore_retn
4507 <1>
4508 <1> copy_source_file_to_destination_pass_q:
4509 000090C2 B002 <1> mov al, 2 ; copy procedure Phase 2
4510 000090C4 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 000090C9 6651 <1> push cx
4515 000090CB BE[99000100] <1> mov esi, msg_copy_nextline
4516 000090D0 E8E8D1FFFF <1> call print_msg
4517 <1> ;pop cx
4518 000090D5 6658 <1> pop ax
4519 <1>
4520 <1> ;or cl, cl
4521 000090D7 08C0 <1> or al, al
4522 000090D9 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 000090DB 3C12 <1> cmp al, 18
4528 000090DD 7506 <1> jne short copy_source_file_to_destination_not_OK
4529 <1> ;
4530 <1> ;mov al, cl ; error number (write fault!)
4531 000090DF F9 <1> stc
4532 000090E0 E99AF2FFFF <1> jmp loc_file_rw_cmd_failed
4533 <1>
4534 <1> copy_source_file_to_destination_not_OK:
4535 000090E5 BE[D4000100] <1> mov esi, Msg_read_file_error_before_EOF
4536 000090EA E8CED1FFFF <1> call print_msg
4537 000090EF E98BF2FFFF <1> jmp loc_file_rw_restore_retn
4538 <1>
4539 <1> copy_source_file_to_destination_OK:

```



```

4540 000090F4 BE[39FF0000] <1>      mov     esi, Msg_OK
4541 000090F9 E8BFD1FFFF <1>      call    print_msg
4542 <1>
4543 000090FE 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>      ;jnb     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 00009103 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 00009104 8A06 <1>      mov     al, [esi]
4571 00009106 3C20 <1>      cmp     al, 20h
4572 00009108 771E <1>      ja      short loc_find_env
4573 <1>
4574 0000910A BE00300900 <1>      mov     esi, Env_Page
4575 <1> loc_print_setline:
4576 0000910F 803E00 <1>      cmp     byte [esi], 0
4577 00009112 7613 <1>      jna      short loc_setenv_retn
4578 00009114 E8A4D1FFFF <1>      call    print_msg
4579 00009119 56 <1>      push     esi
4580 0000911A BE[13070100] <1>      mov     esi, nextline
4581 0000911F E899D1FFFF <1>      call    print_msg
4582 00009124 5E <1>      pop      esi
4583 00009125 EBE8 <1>      jmp     short loc_print_setline
4584 <1>
4585 <1> loc_setenv_retn:
4586 00009127 C3 <1>      retn
4587 <1>
4588 <1> loc_find_env:
4589 00009128 3C3D <1>      cmp     al, '='
4590 0000912A 0F841DE9FFFF <1>      je      loc_cmd_failed
4591 <1>
4592 00009130 56 <1>      push     esi
4593 <1> loc_repeat_env_equal_check:
4594 00009131 46 <1>      inc     esi
4595 00009132 803E3D <1>      cmp     byte [esi], '='
4596 00009135 7431 <1>      je      short pass_env_equal_check
4597 00009137 803E20 <1>      cmp     byte [esi], 20h
4598 0000913A 73F5 <1>      jnb     short loc_repeat_env_equal_check
4599 0000913C C60600 <1>      mov     byte [esi], 0
4600 0000913F 5E <1>      pop      esi
4601 00009140 BF[E6390100] <1>      mov     edi, TextBuffer ; out buffer
4602 00009145 B9FF000000 <1>      mov     ecx, 255 ; maximum size (limit)
4603 0000914A 30C0 <1>      xor     al, al ; 0 -> use [ESI]
4604 0000914C E89E000000 <1>      call    get_environment_string
4605 00009151 72D4 <1>      jc      short loc_setenv_retn
4606 <1>
4607 00009153 BE[E6390100] <1>      mov     esi, TextBuffer
4608 00009158 E860D1FFFF <1>      call    print_msg
4609 0000915D BE[13070100] <1>      mov     esi, nextline
4610 00009162 E856D1FFFF <1>      call    print_msg
4611 <1>
4612 00009167 C3 <1>      retn
4613 <1>
4614 <1> pass_env_equal_check:
4615 00009168 46 <1>      inc     esi
4616 00009169 803E20 <1>      cmp     byte [esi], 20h
4617 0000916C 73FA <1>      jnb     short pass_env_equal_check
4618 0000916E C60600 <1>      mov     byte [esi], 0
4619 <1>
4620 <1> loc_call_set_env_string:
4621 00009171 5E <1>      pop      esi
4622 00009172 E83B010000 <1>      call    set_environment_string
4623 00009177 73AE <1>      jnc     short loc_setenv_retn
4624 <1>
4625 <1> loc_set_cmd_failed:
4626 00009179 3C08 <1>      cmp     al, 08h
4627 0000917B 0F85CCE8FFFF <1>      jne     loc_cmd_failed
4628 <1>
4629 00009181 BE[14010100] <1>      mov     esi, Msg_No_Set_Space
4630 00009186 E832D1FFFF <1>      call    print_msg
4631 <1>
4632 0000918B C3 <1>      retn
4633 <1>
4634 <1> set_get_path:
4635 <1>      ; 11/04/2016 (TRDOS 386 = TRDOS v2.0)
4636 <1>      ; 03/09/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_path')
4637 <1>      ; 2005
4638 <1> get_path_fchar:
4639 <1>      ; esi = path
4640 0000918C 803E20 <1>      cmp     byte [esi], 20h
4641 0000918F 7737 <1>      ja      short loc_set_path

```

```

4642                                     <1>
4643 00009191 BE00300900               <1>      mov     esi, Env_Page
4644                                     <1> loc_print_path:
4645 00009196 803E00                   <1>      cmp     byte [esi], 0
4646 00009199 762C                     <1>      jna     short loc_path_retn
4647                                     <1>
4648 0000919B BE[74FB0000]               <1>      mov     esi, Cmd_Path ; 'PATH' address
4649 000091A0 BF[E6390100]               <1>      mov     edi, TextBuffer ; oout buffer
4650 000091A5 30C0                     <1>      xor     al, al ; use [ESI]
4651 000091A7 B9FF000000               <1>      mov     ecx, 255 ; maximum size (limit)
4652 000091AC E83E000000               <1>      call    get_environment_string
4653 000091B1 7214                     <1>      jc      short loc_path_retn
4654                                     <1>
4655 000091B3 BE[E6390100]               <1>      mov     esi, TextBuffer
4656 000091B8 E800D1FFFF               <1>      call    print_msg
4657 000091BD BE[13070100]               <1>      mov     esi, nextline
4658 000091C2 E8F6D0FFFF               <1>      call    print_msg
4659                                     <1>
4660                                     <1> loc_path_retn:
4661 000091C7 C3                       <1>      retn
4662                                     <1>
4663                                     <1> loc_set_path:
4664 000091C8 56                       <1>      push    esi
4665                                     <1> loc_set_path_find_end:
4666 000091C9 46                       <1>      inc     esi
4667 000091CA 803E20                   <1>      cmp     byte [esi], 20h
4668 000091CD 73FA                     <1>      jnb     short loc_set_path_find_end
4669 000091CF C60600                   <1>      mov     byte [esi], 0
4670                                     <1> loc_set_path_header:
4671 000091D2 5E                       <1>      pop     esi
4672 000091D3 4E                       <1>      dec     esi
4673 000091D4 C6063D                   <1>      mov     byte [esi], '='
4674 000091D7 4E                       <1>      dec     esi
4675 000091D8 C60648                   <1>      mov     byte [esi], 'H'
4676 000091DB 4E                       <1>      dec     esi
4677 000091DC C60654                   <1>      mov     byte [esi], 'T'
4678 000091DF 4E                       <1>      dec     esi
4679 000091E0 C60641                   <1>      mov     byte [esi], 'A'
4680 000091E3 4E                       <1>      dec     esi
4681 000091E4 C60650                   <1>      mov     byte [esi], 'P'
4682                                     <1>
4683                                     <1> loc_path_call_set_env_string:
4684 000091E7 E8C6000000               <1>      call    set_environment_string
4685 000091EC 728B                     <1>      jc      short loc_set_cmd_failed
4686                                     <1>
4687 000091EE C3                       <1>      retn
4688                                     <1>
4689                                     <1> get_environment_string:
4690                                     <1>      ; 12/04/2016
4691                                     <1>      ; 11/04/2016
4692                                     <1>      ; 05/04/2016 (TRDOS 386 = TRDOS v2.0)
4693                                     <1>      ; 02/09/2011 (TRDOS v1, MAINPROG.ASM)
4694                                     <1>      ; 28/08/2011
4695                                     <1>      ; INPUT->
4696                                     <1>      ;     EDI = Output buffer
4697                                     <1>      ;     CX = Buffer length (<= ENV_PAGE_SIZE)
4698                                     <1>      ;
4699                                     <1>      ;     AL > 0 = AL = String sequence number
4700                                     <1>      ;     AL = 0 -> ESI = ASCIIZ Set word
4701                                     <1>      ;     (environment variable)
4702                                     <1>      ; OUTPUT ->
4703                                     <1>      ;     ESI is not changed
4704                                     <1>      ;     EDI is not changed
4705                                     <1>      ;     EAX = String length (with zero tail)
4706                                     <1>      ;     EDX = Environment variables page address
4707                                     <1>      ;     CF = 1 -> Not found (EAX not valid)
4708                                     <1>      ;
4709                                     <1>      ; (Modified registers: EAX, EDX)
4710                                     <1>
4711 000091EF BA00300900               <1>      mov     edx, Env_Page
4712 000091F4 803A00                   <1>      cmp     byte [edx], 0
4713 000091F7 7474                     <1>      jz      short get_env_string_with_word_stc_retn
4714                                     <1>
4715 000091F9 66890D[54450100]           <1>      mov     [env_var_length], cx
4716                                     <1>
4717 00009200 51                       <1>      push    ecx ; *
4718 00009201 56                       <1>      push    esi ; **
4719                                     <1>
4720 00009202 08C0                     <1>      or      al, al
4721 00009204 7449                     <1>      jz      short get_env_string_with_word
4722                                     <1>
4723                                     <1> get_env_string_with_seq_number:
4724 00009206 B101                     <1>      mov     cl, 1
4725 00009208 88C5                     <1>      mov     ch, al
4726 0000920A 31C0                     <1>      xor     eax, eax
4727 0000920C 89D6                     <1>      mov     esi, edx ; Env_Page
4728                                     <1>
4729                                     <1> get_env_string_seq_number_check:
4730 0000920E 38CD                     <1>      cmp     ch, cl
4731 00009210 7726                     <1>      ja      short get_env_string_seq_number_next
4732                                     <1>
4733                                     <1> get_env_string_move_to_buff:
4734 00009212 57                       <1>      push    edi ; ***
4735                                     <1>
4736 00009213 29D2                     <1>      sub     edx, edx
4737                                     <1>
4738                                     <1> get_env_string_seq_number_repeat1:
4739 00009215 42                       <1>      inc     edx
4740 00009216 AC                       <1>      lodsb
4741 00009217 AA                       <1>      stosb
4742                                     <1>
4743 00009218 66FF0D[54450100]           <1>      dec     word [env_var_length]

```

```

4744 0000921F 7508      <1>      jnz      short get_env_string_seq_number_repeat3
4745
4746      <1> get_env_string_seq_number_repeat2:
4747      <1>      and      al, al
4748 00009223 7408      <1>      jz       short get_env_string_seq_number_ok
4749 00009225 42        <1>      inc      edx
4750 00009226 AC        <1>      lodsb
4751 00009227 EBF8      <1>      jmp      short get_env_string_seq_number_repeat2
4752
4753      <1> get_env_string_seq_number_repeat3:
4754 00009229 08C0      <1>      or       al, al
4755 0000922B 75E8      <1>      jnz      short get_env_string_seq_number_repeat1
4756
4757      <1> get_env_string_seq_number_ok:
4758 0000922D 5F        <1>      pop      edi ; ***
4759 0000922E 89D0      <1>      mov      eax, edx ; Length of the environment string
4760                                ; (ASCIIIZ, includes ZERO tail)
4761 00009230 BA00300900 <1>      mov      edx, Env_Page
4762
4763      <1> get_env_string_stc_retn:
4764 00009235 5E        <1>      pop      esi ; **
4765 00009236 59        <1>      pop      ecx ; *
4766 00009237 C3        <1>      retn
4767
4768      <1> get_env_string_seq_number_next:
4769 00009238 AC        <1>      lodsb
4770 00009239 08C0      <1>      or       al, al
4771 0000923B 75FB      <1>      jnz      short get_env_string_seq_number_next
4772
4773 0000923D 81FE00320900 <1>      cmp      esi, Env_Page + Env_Page_Size ; +512 (+4096)
4774 00009243 F5        <1>      cmc
4775 00009244 72EF      <1>      jc       short get_env_string_stc_retn
4776
4777 00009246 AC        <1>      lodsb
4778 00009247 3C01      <1>      cmp      al, 1
4779 00009249 72EA      <1>      jb       short get_env_string_stc_retn
4780 0000924B FEC1      <1>      inc      cl
4781 0000924D EBBF      <1>      jmp      short get_env_string_seq_number_check
4782
4783      <1> get_env_string_with_word:
4784 0000924F 31C9      <1>      xor      ecx, ecx
4785
4786      <1> get_env_string_calc_word_length:
4787 00009251 AC        <1>      lodsb
4788 00009252 3C20      <1>      cmp      al, 20h
4789 00009254 7211      <1>      jb       short get_env_string_calc_word_length_ok
4790                                ;inc      cx
4791 00009256 FEC1      <1>      inc      cl
4792
4793 00009258 3C61      <1>      cmp      al, 'a'
4794 0000925A 72F5      <1>      jb       short get_env_string_calc_word_length
4795 0000925C 3C7A      <1>      cmp      al, 'z'
4796 0000925E 77F1      <1>      ja       short get_env_string_calc_word_length
4797 00009260 24DF      <1>      and      al, 0DFh
4798 00009262 8846FF      <1>      mov      [esi-1], al
4799 00009265 EBFA      <1>      jmp      short get_env_string_calc_word_length
4800
4801      <1> get_env_string_calc_word_length_ok:
4802 00009267 08C9      <1>      or       cl, cl
4803 00009269 7506      <1>      jnz      short get_env_string_calc_word_length_save
4804
4805 0000926B 5E        <1>      pop      esi ; **
4806
4807      <1> get_env_string_stc_retn1:
4808 0000926C 59        <1>      pop      ecx ; *
4809
4810      <1> get_env_string_with_word_stc_retn:
4811 0000926D 31C0      <1>      xor      eax, eax
4812 0000926F F9        <1>      stc
4813 00009270 C3        <1>      retn
4814
4815      <1> get_env_string_calc_word_length_save:
4816 00009271 871C24      <1>      xchg     ebx, [esp] ; **
4817 00009274 89DE      <1>      mov      esi, ebx
4818                                ; Start of the env string (to be searched)
4819
4820 00009276 57        <1>      push     edi ; ***
4821 00009277 89D7      <1>      mov      edi, edx ; Env_Page
4822
4823      <1> get_env_string_compare:
4824 00009279 57        <1>      push     edi ; ****
4825 0000927A 51        <1>      push     ecx ; ***** ; Variable name length
4826
4827      <1> get_env_string_compare_rep:
4828 0000927B AC        <1>      lodsb
4829 0000927C AE        <1>      scasb
4830 0000927D 7511      <1>      jne      short get_env_string_compare_next1
4831 0000927F E2FA      <1>      loop     get_env_string_compare_rep
4832
4833 00009281 803F3D      <1>      cmp      byte [edi], '='
4834 00009284 750A      <1>      jne      short get_env_string_compare_next1
4835
4836 00009286 59        <1>      pop      ecx ; *****
4837 00009287 5F        <1>      pop      edi ; ****
4838 00009288 89FE      <1>      mov      esi, edi
4839 0000928A 5F        <1>      pop      edi ; ***
4840 0000928B 871C24      <1>      xchg     ebx, [esp] ; **
4841 0000928E EB82      <1>      jmp      short get_env_string_move_to_buff
4842
4843      <1> get_env_string_compare_next1:
4844 00009290 89FE      <1>      mov      esi, edi
4845 00009292 59        <1>      pop      ecx ; *****

```

```

4846 00009293 5F      <1>      pop     edi ; ****
4847                  <1> get_env_string_compare_next2:
4848 00009294 81FEFF310900 <1>      cmp     esi, Env_Page + Env_Page_Size - 1 ; +511 (+4095)
4849 0000929A 7310      <1>      jnb     short get_env_string_compare_not_ok
4850 0000929C 20C0      <1>      and     al, al
4851 0000929E AC        <1>      lodsb
4852 0000929F 75F3      <1>      jnz     short get_env_string_compare_next2
4853 000092A1 08C0      <1>      or      al, al
4854 000092A3 7407      <1>      jz      short get_env_string_compare_not_ok
4855 000092A5 4E        <1>      dec     esi ; 12/04/2016
4856 000092A6 89F7      <1>      mov     edi, esi
4857 000092A8 89DE      <1>      mov     esi, ebx
4858 000092AA EB CD      <1>      jmp     short get_env_string_compare
4859                  <1>
4860                  <1> get_env_string_compare_not_ok:
4861 000092AC 5F        <1>      pop     edi ; ***
4862 000092AD 89DE      <1>      mov     esi, ebx
4863 000092AF 5B        <1>      pop     ebx ; **
4864 000092B0 EB BA      <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 000092B2 56        <1>      push    esi ; *
4888                  <1>
4889 000092B3 31C0      <1>      xor     eax, eax
4890                  <1>
4891                  <1> set_env_chk_validation1:
4892 000092B5 FEC4      <1>      inc     ah ; variable (string) length
4893 000092B7 AC        <1>      lodsb
4894 000092B8 3C3D      <1>      cmp     al, '='
4895 000092BA 7415      <1>      je      short set_env_chk_validation2
4896 000092BC 3C20      <1>      cmp     al, 20h
4897 000092BE 720F      <1>      jb      short set_env_string_stc
4898                  <1>
4899                  <1>      ; 06/04/2016
4900 000092C0 3C61      <1>      cmp     al, 'a'
4901 000092C2 72F1      <1>      jb      short set_env_chk_validation1
4902 000092C4 3C7A      <1>      cmp     al, 'z'
4903 000092C6 77ED      <1>      ja      short set_env_chk_validation1
4904 000092C8 2C20      <1>      sub     al, 'a'-'A'
4905 000092CA 8846FF      <1>      mov     [esi-1], al
4906 000092CD EBE6      <1>      jmp     short set_env_chk_validation1
4907                  <1>
4908                  <1> set_env_string_stc:
4909 000092CF 5E        <1>      pop     esi ; *
4910                  <1>      ;stc
4911 000092D0 C3        <1>      retn
4912                  <1>
4913                  <1> set_env_chk_validation2:
4914 000092D1 51        <1>      push    ecx ; **
4915 000092D2 53        <1>      push    ebx ; ***
4916 000092D3 57        <1>      push    edi ; ****
4917                  <1>
4918                  <1>      ; 12/04/2016
4919 000092D4 8B5C240C <1>      mov     ebx, [esp+12]
4920                  <1>
4921                  <1> set_env_chk_validation2w:
4922 000092D8 89F7      <1>      mov     edi, esi
4923 000092DA 4F        <1>      dec     edi
4924                  <1>
4925 000092DB 807FFF20 <1>      cmp     byte [edi-1], 20h
4926 000092DF 771A      <1>      ja      short set_env_chk_validation2z
4927                  <1>
4928 000092E1 56        <1>      push    esi
4929 000092E2 89FE      <1>      mov     esi, edi
4930 000092E4 4E        <1>      dec     esi
4931                  <1>
4932                  <1> set_env_chk_validation2x:
4933 000092E5 4E        <1>      dec     esi
4934                  <1>
4935 000092E6 39DE      <1>      cmp     esi, ebx
4936 000092E8 7207      <1>      jb      short set_env_chk_validation2y
4937                  <1>
4938 000092EA 4F        <1>      dec     edi
4939                  <1>
4940 000092EB 8A06      <1>      mov     al, [esi]
4941 000092ED 8807      <1>      mov     [edi], al
4942                  <1>
4943 000092EF EBF4      <1>      jmp     short set_env_chk_validation2x
4944                  <1>
4945                  <1> set_env_chk_validation2y:
4946 000092F1 5E        <1>      pop     esi
4947                  <1>

```



```

4948      <1>      ;mov    byte [ebx], 20h
4949      <1>
4950 000092F2 43      <1>      inc     ebx
4951 000092F3 895C240C <1>      mov     [esp+12], ebx
4952      <1>
4953 000092F7 FECC     <1>      dec     ah ; 13/04/2016
4954      <1>
4955 000092F9 EBDD     <1>      jmp     short set_env_chk_validation2w
4956      <1>
4957      <1> set_env_chk_validation2z:
4958 000092FB BA00300900 <1>      mov     edx, Env_Page
4959 00009300 89D7     <1>      mov     edi, edx
4960      <1>
4961      <1> set_env_chk_validation3:
4962 00009302 AC      <1>      lodsb
4963 00009303 3C20     <1>      cmp     al, 20h
4964 00009305 74FB     <1>      je      short set_env_chk_validation3
4965      <1>
4966 00009307 9C      <1>      pushf
4967      <1>
4968      <1>      ; 12/04/2016
4969      <1> set_env_chk_validation3n:
4970 00009308 3C61     <1>      cmp     al, 'a'
4971 0000930A 720C     <1>      jnb     short set_env_chk_validation3c
4972 0000930C 3C7A     <1>      cmp     al, 'z'
4973 0000930E 7705     <1>      ja      short set_env_chk_validation3x
4974 00009310 2C20     <1>      sub     al, 'a'-'A'
4975 00009312 8846FF     <1>      mov     [esi-1], al
4976      <1>
4977      <1> set_env_chk_validation3x:
4978 00009315 AC      <1>      lodsb
4979 00009316 EBF0     <1>      jmp     short set_env_chk_validation3n
4980      <1>
4981      <1> set_env_chk_validation3c:
4982 00009318 3C20     <1>      cmp     al, 20h
4983 0000931A 73F9     <1>      jnb     short set_env_chk_validation3x
4984      <1>
4985 0000931C 803F00     <1>      cmp     byte [edi], 0
4986 0000931F 7731     <1>      ja      short set_env_chk_validation4
4987      <1>
4988 00009321 9D      <1>      popf
4989 00009322 7228     <1>      jb      short set_env_string_nothing
4990      <1>
4991 00009324 B900020000 <1>      mov     ecx, Env_Page_Size ; 512 (4096)
4992      <1>
4993 00009329 89DE     <1>      mov     esi, ebx ; 12/04/2016
4994      <1>
4995      <1> set_env_string_copy_to_envb:
4996 0000932B AC      <1>      lodsb
4997 0000932C 3C20     <1>      cmp     al, 20h
4998 0000932E 720A     <1>      jnb     short set_env_string_copy_to_envb_z
4999 00009330 AA      <1>      stosb
5000 00009331 E2F8     <1>      loop    set_env_string_copy_to_envb
5001      <1>
5002      <1>      ; 11/04/2016
5003 00009333 89D7     <1>      mov     edi, edx ; Env_Page
5004 00009335 B900020000 <1>      mov     ecx, Env_Page_Size
5005      <1>
5006      <1> set_env_string_copy_to_envb_z:
5007 0000933A 52      <1>      push    edx ; Start address of the variable
5008 0000933B BA00020000 <1>      mov     edx, Env_Page_Size
5009 00009340 29CA     <1>      sub     edx, ecx ; variable (string) length
5010      <1>
5011 00009342 28C0     <1>      sub     al, al ; 0
5012 00009344 F3AA     <1>      rep     stosb ; clear remain bytes of the env page
5013      <1>
5014 00009346 58      <1>      pop     eax ; Start address of the variable
5015      <1>
5016      <1> set_env_string_allocate_envb_retn: ; stc or clc return
5017 00009347 5F      <1>      pop     edi ; ****
5018 00009348 5B      <1>      pop     ebx ; ***
5019 00009349 59      <1>      pop     ecx ; **
5020 0000934A 5E      <1>      pop     esi ; *
5021 0000934B C3      <1>      retn
5022      <1>
5023      <1> set_env_string_nothing:
5024 0000934C 31C0     <1>      xor     eax, eax
5025 0000934E 31D2     <1>      xor     edx, edx ; 11/04/2016
5026 00009350 EBF5     <1>      jmp     short set_env_string_allocate_envb_retn
5027      <1>
5028      <1> set_env_chk_validation4:
5029      <1>      ; 11/04/2016
5030 00009352 9D      <1>      popf
5031      <1>
5032 00009353 89D6     <1>      mov     esi, edx ; Env_Page
5033      <1>
5034      <1> set_env_chk_validation5:
5035 00009355 89DF     <1>      mov     edi, ebx ; ASCIIIZ environment string address
5036 00009357 0FB6CC     <1>      movzx   ecx, ah ; Variable (string) length (with '=')
5037      <1>
5038      <1> set_env_chk_validation5_loop:
5039 0000935A AC      <1>      lodsb
5040 0000935B AE      <1>      scasb
5041 0000935C 750A     <1>      jne     short set_env_chk_validation6
5042 0000935E E2FA     <1>      loop    set_env_chk_validation5_loop
5043      <1>
5044 00009360 3C3D     <1>      cmp     al, '='
5045 00009362 0F8483000000 <1>      je      set_env_change_variable
5046      <1>
5047      <1> set_env_chk_validation6:
5048 00009368 08C0     <1>      or      al, al ; 0
5049 0000936A 7403     <1>      jz      short set_env_chk_validation7

```

```

5050
5051 0000936C AC
5052 0000936D EBF9
5053
5054
5055 0000936F 88E1
5056 00009371 01F1
5057 00009373 81F9FF310900
5058
5059
5060 00009379 72DA
5061
5062
5063 0000937B 0FB6F4
5064 0000937E 01DE
5065
5066
5067 00009380 AC
5068 00009381 3C20
5069 00009383 74FB
5070 00009385 72C5
5071
5072
5073 00009387 AC
5074 00009388 3C20
5075 0000938A 73FB
5076
5077
5078
5079
5080 0000938C 29DE
5081
5082 0000938E 56
5083
5084 0000938F 89D6
5085
5086 00009391 B900020000
5087
5088
5089 00009396 AC
5090 00009397 20C0
5091 00009399 7406
5092 0000939B E2F9
5093
5094
5095 0000939D 884EFF
5096 000093A0 41
5097
5098
5099 000093A1 49
5100 000093A2 7408
5101 000093A4 AC
5102 000093A5 08C0
5103 000093A7 740C
5104 000093A9 49
5105 000093AA 75EA
5106
5107
5108 000093AC 58
5109 000093AD B808000000
5110 000093B2 F9
5111 000093B3 EB92
5112
5113
5114 000093B5 8B0C24
5115 000093B8 4E
5116 000093B9 89F0
5117 000093BB 01C8
5118 000093BD 81C200020000
5119 000093C3 39D0
5120 000093C5 77E5
5121 000093C7 49
5122 000093C8 89CA
5123 000093CA 89F7
5124 000093CC 893C24
5125 000093CF 89DE
5126 000093D1 F3A4
5127 000093D3 28C0
5128 000093D5 AA
5129 000093D6 58
5130 000093D7 81FF00320900
5131 000093DD 0F8364FFFFFF
5132 000093E3 880F
5133 000093E5 F8
5134 000093E6 E95CFFFFFF
5135
5136
5137
5138
5139
5140
5141
5142 000093EB 8825[54450100]
5143
5144 000093F1 28C9
5145
5146 000093F3 57
5147
5148 000093F4 89F7
5149
5150
5151 000093F6 AC

```

```

<1>
<1>      lodsb
<1>      jmp      short set_env_chk_validation6
<1>
<1> set_env_chk_validation7:
<1>      mov      cl, ah
<1>      add      ecx, esi
<1>      cmp      ecx, Env_Page + Env_Page_Size - 1
<1>              ; 511 (4095)
<1>              ; strlen + '=' + 0
<1>      jnb      short set_env_chk_validation5
<1>
<1> set_env_chk_validation8: ; variable not found
<1>      movzx    esi, ah ; variable name length (with '=')
<1>      add      esi, ebx ; position just after of the '='
<1>
<1> set_env_chk_validation8_loop:
<1>      lodsb
<1>      cmp      al, 20h
<1>      je       short set_env_chk_validation8_loop
<1>      jnb      short set_env_string_nothing
<1>
<1> set_env_chk_validation9:
<1>      lodsb
<1>      cmp      al, 20h
<1>      jnb      short set_env_chk_validation9
<1>
<1>      ; End of ASCIIIZ environment string
<1>
<1> set_env_add_variable:
<1>      sub      esi, ebx ; variable+definition length
<1>
<1>      push     esi ; *****
<1>
<1>      mov      esi, edx ; Environment page address
<1>
<1>      mov      ecx, Env_Page_Size ; 512 (4096)
<1>
<1> set_env_add_variable_loop:
<1>      lodsb
<1>      and      al, al
<1>      jz       short set_env_add_variable_chk1 ; 0
<1>      loop     set_env_add_variable_loop
<1>
<1>      ; 11/04/2016
<1>      mov      [esi-1], cl ; 0
<1>      inc      ecx
<1>
<1> set_env_add_variable_chk1:
<1>      dec      ecx
<1>      jz       short set_env_add_variable_nspc
<1>      lodsb
<1>      or       al, al
<1>      jz       short set_env_add_variable_chk2 ; 00
<1>      dec      ecx
<1>      jnz      short set_env_add_variable_loop
<1>
<1> set_env_add_variable_nspc: ; no space on environment page
<1>      pop      eax ; *****
<1>      mov      eax, 8 ; No space for new environment string
<1>      stc
<1>      jmp      short set_env_string_allocate_envb_retn
<1>
<1> set_env_add_variable_chk2:
<1>      mov      ecx, [esp] ; *****
<1>      dec      esi ; beginning address of the new variable
<1>      mov      eax, esi
<1>      add      eax, ecx ; string length (with CR)
<1>      add      edx, Env_Page_Size ; 512 (4096)
<1>      cmp      eax, edx
<1>      ja       short set_env_add_variable_nspc
<1>      dec      ecx ; except CR at the end
<1>      mov      edx, ecx ; 12/04/2016
<1>      mov      edi, esi
<1>      mov      [esp], edi ; ***** ; Start address of new variable
<1>      mov      esi, ebx ; ASCIIIZ environment string address
<1>      rep      movsb
<1>      sub      al, al
<1>      stosb
<1>      pop      eax ; ***** ; Beginning address of new variable
<1>      cmp      edi, Env_Page + Env_Page_Size ; 12/04/2016
<1>      jnb      set_env_string_allocate_envb_retn ; OK !
<1>      mov      [edi], cl ; 0
<1>      cld      ; 13/04/2016
<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 = ASCIIIZ environment string address (after '=')
<1>
<1>      ; ah = variable length from start to the '='
<1>      mov      [env_var_length], ah
<1>
<1>      sub      cl, cl ; ecx = 0
<1>
<1>      push     edi ; *****
<1>
<1>      mov      edi, esi ; 11/04/2016
<1>
<1> set_env_change_variable_calcl:
<1>      lodsb

```

```

5152 000093F7 08C0      <1>      or      al, al
5153 000093F9 7403      <1>      jz       short set_env_change_variable_calc2
5154                      <1>
5155 000093FB 41          <1>      inc      ecx ; length of environment string (after the '=')
5156                      <1>
5157 000093FC EBF8      <1>      jmp      short set_env_change_variable_calc1
5158                      <1>
5159                      <1> set_env_change_variable_calc2:
5160 000093FE 8B3424      <1>      mov      esi, [esp] ; ASCIIZ environment string address
5161                      <1>
5162 00009401 29D2      <1>      sub      edx, edx
5163                      <1>
5164                      <1> set_env_change_variable_calc3:
5165 00009403 AC        <1>      lodsb
5166 00009404 3C20      <1>      cmp      al, 20h
5167 00009406 7203      <1>      jb       short set_env_change_variable_calc4
5168                      <1>
5169 00009408 42          <1>      inc      edx ; length of ASCIIZ string (after the '=')
5170                      <1>
5171 00009409 EBF8      <1>      jmp      short set_env_change_variable_calc3
5172                      <1>
5173                      <1> set_env_change_variable_calc4:
5174 0000940B C646FF00 <1>      mov      byte [esi-1], 0 ; put ZERO instead of CR
5175                      <1>
5176 0000940F 5E          <1>      pop      esi ; ***** ; ASCIIZ string address (after '=')
5177                      <1>
5178                      <1> ; EDI = Old variable's address (after '=')
5179                      <1>
5180                      <1> ; compare the new string with the old string
5181 00009410 39CA      <1>      cmp      edx, ecx
5182 00009412 7717      <1>      ja       short set_env_change_variable_calc5 ; longer
5183 00009414 0F828F000000 <1>      jb       set_env_change_variable_calc9 ; shorter
5184                      <1>
5185                      <1> ;same length (simple copy)
5186 0000941A 0FB6C4      <1>      movzx    eax, ah
5187 0000941D 01C2      <1>      add      edx, eax
5188 0000941F F7D8      <1>      neg      eax
5189 00009421 01F8      <1>      add      eax, edi
5190                      <1> ; EAX = Start address of the variable
5191                      <1> ; EDX = Variable length (without ZERO at the end of variable)
5192                      <1>
5193 00009423 F3A4      <1>      rep      movsb
5194 00009425 F8          <1>      cld      ; 13/04/2016
5195 00009426 E91CFFFFFF <1>      jmp      set_env_string_allocate_envb_retn ; OK !
5196                      <1>
5197                      <1> set_env_change_variable_calc5:
5198                      <1> ; 11/04/2016
5199 0000942B 52          <1>      push    edx ; *****
5200 0000942C 29CA      <1>      sub      edx, ecx ; difference ; (the new string is longer)
5201 0000942E 89F3      <1>      mov      ebx, esi
5202 00009430 89FE      <1>      mov      esi, edi
5203                      <1>
5204                      <1> set_env_change_variable_calc6:
5205 00009432 AC        <1>      lodsb
5206 00009433 20C0      <1>      and      al, al
5207 00009435 75FB      <1>      jnz      short set_env_change_variable_calc6
5208                      <1>
5209 00009437 81FE00320900 <1>      cmp      esi, Env_Page + Env_Page_Size ; 512 (4096)
5210 0000943D 0F8369FFFFFF <1>      jnb      set_env_add_variable_nspc
5211                      <1>
5212 00009443 89F9      <1>      mov      ecx, edi ; current (old) variable's address
5213 00009445 89F7      <1>      mov      edi, esi ; next variable's address
5214                      <1>
5215 00009447 AC        <1>      lodsb
5216 00009448 08C0      <1>      or       al, al
5217 0000944A 7416      <1>      jz       short set_env_change_variable_calc8 ; 00
5218                      <1>
5219                      <1> set_env_change_variable_calc7:
5220 0000944C AC        <1>      lodsb
5221 0000944D 20C0      <1>      and      al, al
5222 0000944F 75FB      <1>      jnz      short set_env_change_variable_calc7
5223                      <1>
5224 00009451 81FE00320900 <1>      cmp      esi, Env_Page + Env_Page_Size ; 512 (4096)
5225 00009457 0F834FFFFFFF <1>      jnb      set_env_add_variable_nspc
5226                      <1>
5227 0000945D AC        <1>      lodsb
5228 0000945E 08C0      <1>      or       al, al
5229 00009460 75EA      <1>      jnz      short set_env_change_variable_calc7
5230                      <1>
5231                      <1> set_env_change_variable_calc8:
5232 00009462 4E          <1>      dec      esi ; address of the second (last) 0 of the 00
5233                      <1>
5234 00009463 01F2      <1>      add      edx, esi ; final position of the last 0
5235                      <1>
5236 00009465 81FA00320900 <1>      cmp      edx, Env_Page + Env_Page_Size ; 512 (4096)
5237 0000946B 0F833BFFFFFF <1>      jnb      set_env_add_variable_nspc
5238                      <1>
5239 00009471 89C8      <1>      mov      eax, ecx ; old variable's address (after '=')
5240                      <1>
5241 00009473 89F1      <1>      mov      ecx, esi
5242 00009475 29F9      <1>      sub      ecx, edi ; count of bytes to move forward
5243                      <1>
5244                      <1> ; 13/04/2016
5245 00009477 C60200 <1>      mov      byte [edx], 0
5246 0000947A 89D7      <1>      mov      edi, edx
5247 0000947C 29F2      <1>      sub      edx, esi ; difference (additional byte count)
5248 0000947E 4F          <1>      dec      edi ; the last zero address (first byte of the 00)
5249 0000947F 89FE      <1>      mov      esi, edi
5250 00009481 29D6      <1>      sub      esi, edx ; - displacement
5251                      <1>
5252 00009483 FA          <1>      cli      ; disable interrupts
5253 00009484 FD          <1>      std      ; backward

```

```

5254                                     <1>
5255 00009485 F3A4                       <1>      rep    movsb ; move ECX bytes from DS:ESI to ES:EDI
5256                                     <1>
5257 00009487 FC                         <1>      cld      ; forward (default)
5258 00009488 FB                         <1>      sti      ; enable interrupts
5259                                     <1>
5260 00009489 89C7                       <1>      mov     edi, eax
5261 0000948B 59                         <1>      pop     ecx ; ***** ; byte count (after '=')
5262 0000948C 89CA                       <1>      mov     edx, ecx
5263 0000948E 89DE                       <1>      mov     esi, ebx ; ASCIIIZ string address (after '=')
5264 00009490 89FB                       <1>      mov     ebx, edi
5265                                     <1>
5266 00009492 F3A4                       <1>      rep    movsb
5267                                     <1>
5268 00009494 880F                       <1>      mov     [edi], cl ; 0 ; end of variable
5269                                     <1>
5270 00009496 0FB605[54450100]           <1>      movzx   eax, byte [env_var_length]
5271 0000949D 01C2                       <1>      add     edx, eax ; variable length (total)
5272 0000949F F7D8                       <1>      neg     eax
5273 000094A1 01D8                       <1>      add     eax, ebx ; start address of the variable
5274 000094A3 F8                         <1>      clc      ; 13/04/2016
5275 000094A4 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 000094A9 21D2                       <1>      and     edx, edx ; is empty ?
5280 000094AB 753B                       <1>      jnz     short set_env_change_variable_calc15
5281                                     <1>
5282 000094AD 0FB6DC                       <1>      movzx   ebx, ah
5283 000094B0 F7DB                       <1>      neg     ebx
5284 000094B2 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 000094B4 89FE                       <1>      mov     esi, edi
5290                                     <1>
5291                                     <1> set_env_change_variable_calc10:
5292 000094B6 AC                         <1>      lodsb
5293 000094B7 08C0                       <1>      or      al, al
5294 000094B9 75FB                       <1>      jnz     short set_env_change_variable_calc10
5295                                     <1>
5296 000094BB B9FF310900                 <1>      mov     ecx, Env_Page + Env_Page_Size - 1
5297                                     <1>
5298 000094C0 39CE                       <1>      cmp     esi, ecx ; +511 (+4095)
5299 000094C2 7604                       <1>      jna     short set_env_change_variable_calc11
5300                                     <1>
5301 000094C4 89CE                       <1>      mov     esi, ecx
5302 000094C6 8806                       <1>      mov     [esi], al ; 0
5303                                     <1>
5304                                     <1> set_env_change_variable_calc11:
5305 000094C8 89DF                       <1>      mov     edi, ebx ; old variable's start address
5306                                     <1>
5307                                     <1> set_env_change_variable_calc12:
5308 000094CA AC                         <1>      lodsb
5309 000094CB AA                         <1>      stosb
5310 000094CC 20C0                       <1>      and     al, al
5311 000094CE 75FA                       <1>      jnz     short set_env_change_variable_calc12
5312 000094D0 39CE                       <1>      cmp     esi, ecx
5313 000094D2 7706                       <1>      ja      short set_env_change_variable_calc13
5314 000094D4 AC                         <1>      lodsb
5315 000094D5 AA                         <1>      stosb
5316 000094D6 20C0                       <1>      and     al, al
5317 000094D8 75F0                       <1>      jnz     short set_env_change_variable_calc12
5318                                     <1>
5319                                     <1> set_env_change_variable_calc13:
5320 000094DA 29F9                       <1>      sub     ecx, edi
5321 000094DC 7203                       <1>      jnb     short set_env_change_variable_calc14
5322 000094DE 41                         <1>      inc     ecx ; 1-512 (1-4096)
5323 000094DF F3AA                       <1>      rep     stosb ; al = 0
5324                                     <1>
5325                                     <1> set_env_change_variable_calc14:
5326 000094E1 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 000094E3 E95FFFEFFF               <1>      jmp     set_env_string_allocate_envb_retn ; OK !
5331                                     <1>
5332                                     <1> set_env_change_variable_calc15:
5333 000094E8 52                         <1>      push    edx ; *****
5334 000094E9 F7DA                       <1>      neg     edx
5335 000094EB 01CA                       <1>      add     edx, ecx ; difference (the old string is longer)
5336 000094ED 89F3                       <1>      mov     ebx, esi
5337 000094EF 89FE                       <1>      mov     esi, edi
5338                                     <1>
5339                                     <1> set_env_change_variable_calc16:
5340 000094F1 AC                         <1>      lodsb
5341 000094F2 20C0                       <1>      and     al, al
5342 000094F4 75FB                       <1>      jnz     short set_env_change_variable_calc16
5343                                     <1>
5344 000094F6 B900320900                 <1>      mov     ecx, Env_Page + Env_Page_Size
5345                                     <1>
5346 000094FB 39CE                       <1>      cmp     esi, ecx ; +512 (+4096)
5347 000094FD 7605                       <1>      jna     short set_env_change_variable_calc17
5348                                     <1>
5349 000094FF 89CE                       <1>      mov     esi, ecx
5350 00009501 8846FF                       <1>      mov     [esi-1], al ; 0
5351                                     <1>
5352                                     <1> set_env_change_variable_calc17:
5353 00009504 89F9                       <1>      mov     ecx, edi ; current (old) variable's address
5354 00009506 89F7                       <1>      mov     edi, esi ; next variable's address
5355                                     <1>

```



```

5356 00009508 AC          <1>      lodsb
5357 00009509 08C0        <1>      or      al, al
5358 0000950B 741D        <1>      jz      short set_env_change_variable_calc20
5359                                <1>
5360                                <1> set_env_change_variable_calc18:
5361 0000950D AC          <1>      lodsb
5362 0000950E 20C0        <1>      and     al, al
5363 00009510 75FB        <1>      jnz     short set_env_change_variable_calc18
5364                                <1>
5365 00009512 81FE00320900 <1>      cmp     esi, Env_Page + Env_Page_Size
5366 00009518 720B        <1>      jnb     short set_env_change_variable_calc19
5367 0000951A 740E        <1>      je      short set_env_change_variable_calc20
5368                                <1>
5369 0000951C BEFF310900 <1>      mov     esi, Env_Page + Env_Page_Size - 1
5370 00009521 8806        <1>      mov     [esi], al ; 0
5371 00009523 EB06        <1>      jmp     short set_env_change_variable_calc21
5372                                <1>
5373                                <1> set_env_change_variable_calc19:
5374 00009525 AC          <1>      lodsb
5375 00009526 08C0        <1>      or      al, al
5376 00009528 75E3        <1>      jnz     short set_env_change_variable_calc18
5377                                <1>
5378                                <1> set_env_change_variable_calc20:
5379 0000952A 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 0000952B 89C8        <1>      mov     eax, ecx ; old variable's address (after '=')
5385                                <1>
5386 0000952D 89F1        <1>      mov     ecx, esi
5387 0000952F 29F9        <1>      sub     ecx, edi ; count of bytes to move backward
5388                                <1>
5389 00009531 89FE        <1>      mov     esi, edi ; next variable's address
5390 00009533 29D7        <1>      sub     edi, edx ; (displacement)
5391                                <1>
5392 00009535 F3A4        <1>      rep     movsb
5393                                <1>
5394 00009537 880F        <1>      mov     [edi], cl ; 0 ; 00 ; end of environment variables
5395                                <1>
5396 00009539 89C7        <1>      mov     edi, eax
5397 0000953B 5A          <1>      pop     edx ; ***** ; byte count (after '=')
5398 0000953C 89D1        <1>      mov     ecx, edx
5399 0000953E 89DE        <1>      mov     esi, ebx ; ASCIIZ string address (after '=')
5400 00009540 89FB        <1>      mov     ebx, edi
5401                                <1>
5402 00009542 F3A4        <1>      rep     movsb
5403                                <1>
5404 00009544 880F        <1>      mov     [edi], cl ; 0 ; end of variable
5405                                <1>
5406 00009546 0FB605[54450100] <1>      movzx   eax, byte [env_var_length]
5407 0000954D 01C2        <1>      add     edx, eax ; variable length (total)
5408 0000954F F7D8        <1>      neg     eax
5409 00009551 01D8        <1>      add     eax, ebx ; start address of the variable
5410 00009553 F8          <1>      cld     ; 13/04/2016
5411 00009554 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 00009559 BE[EEFA0000] <1>      mov     esi, MainProgCfgFile
5420 0000955E 66B80018 <1>      mov     ax, 1800h ; Except volume label and dirs
5421 00009562 E8E8E9FFFF <1>      call    find_first_file
5422 00009567 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 00009569 A3[D4380100] <1>      mov     [MainProgCfg_FileSize], eax
5432                                <1>
5433 0000956E 668B5614 <1>      mov     dx, [esi+DirEntry_FstClusHI]
5434 00009572 C1E210        <1>      shl     edx, 16
5435 00009575 668B561A <1>      mov     dx, [esi+DirEntry_FstClusLO]
5436 00009579 8915[08450100] <1>      mov     [csftdf_sf_cluster], edx
5437                                <1>
5438 0000957F 89C1        <1>      mov     ecx, eax
5439 00009581 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 00009583 E89BBEFFFF <1>      call    allocate_memory_block
5448 00009588 7235        <1>      jc      short loc_load_mainprog_cfg_exit
5449                                <1>
5450 0000958A A3[00450100] <1>      mov     [csftdf_sf_mem_addr], eax ; loading address
5451 0000958F 890D[04450100] <1>      mov     [csftdf_sf_mem_bsize], ecx ; block size
5452                                <1>
5453 00009595 31DB        <1>      xor     ebx, ebx
5454                                <1>      ;mov     [csftdf_sf_rbytes], ebx ; 0, reset
5455                                <1>
5456 00009597 8A3D[E6380100] <1>      mov     bh, [Current_Drv] ; [FindFile_Drv]
5457 0000959D BE00010900 <1>      mov     esi, Logical_DOSDisks

```

```

5458 000095A2 01DE      <1>      add     esi, ebx
5459                      <1>
5460 000095A4 8B1D[00450100] <1>      mov     ebx, [csftdf_sf_mem_addr] ; memory block address
5461                      <1>
5462 000095AA 807E0300      <1>      cmp     byte [esi+LD_FATType], 0
5463 000095AE 7710          <1>      ja      short loc_mcfg_load_fat_file
5464                      <1>
5465 000095B0 C705[10450100]0000- <1>      mov     dword [csftdf_r_size], 65536
5465 000095B8 0100          <1>
5466 000095BA E992010000      <1>      jmp     loc_mcfg_load_fs_file
5467                      <1>
5468                      <1> loc_load_mainprog_cfg_exit:
5469 000095BF C3          <1>      retn
5470                      <1>
5471                      <1> loc_mcfg_load_fat_file:
5472 000095C0 0FB74611      <1>      movzx  eax, word [esi+LD_BPB+BytesPerSec]
5473 000095C4 0FB64E13      <1>      movzx  ecx, byte [esi+LD_BPB+SecPerClust]
5474 000095C8 F7E1          <1>      mul     ecx
5475 000095CA A3[10450100]      <1>      mov     [csftdf_r_size], eax
5476                      <1>
5477                      <1> loc_mcfg_load_fat_file_next:
5478 000095CF E813010000      <1>      call   mcfg_read_fat_file_sectors
5479 000095D4 0F82F7000000      <1>      jc      mcfg_deallocate_mem
5480                      <1>
5481 000095DA 09D2          <1>      or      edx, edx ; edx > 0 -> EOF
5482 000095DC 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 000095DE C705[A4450100]- <1>      mov     dword [mainprog_return_addr], loc_mcfg_ci_return_addr
5486 000095E4 [92960000]      <1>
5487                      <1>      ;
5488 000095E8 8B35[00450100]      <1>      mov     esi, [csftdf_sf_mem_addr]
5489 000095EE 8935[D8380100]      <1>      mov     [MainProgCfg_LineOffset], esi
5490                      <1>
5491 000095F4 A1[D4380100]      <1>      mov     eax, [MainProgCfg_FileSize]
5492 000095F9 89C2          <1>      mov     edx, eax
5493 000095FB 01F2          <1>      add     edx, esi
5494                      <1>
5495                      <1> loc_mcfg_process_next_line_check:
5496 000095FD 89C1          <1>      mov     ecx, eax
5497                      <1>
5498 000095FF 803E2A          <1>      cmp     byte [esi], "*" ; Remark sign
5499 00009602 7503          <1>      jne     short loc_mcfg_process_next_line
5500 00009604 46          <1>      inc     esi
5501 00009605 EB17          <1>      jmp     short loc_move_mainprog_cfg_n11
5502                      <1>
5503                      <1> loc_mcfg_process_next_line:
5504 00009607 83F94F          <1>      cmp     ecx, 79
5505 0000960A 7605          <1>      jna     short loc_start_mainprog_cfg_process
5506                      <1>
5507 0000960C B94F000000      <1>      mov     ecx, 79
5508                      <1>
5509                      <1> loc_start_mainprog_cfg_process:
5510 00009611 BF[96390100] <1>      mov     edi, CommandBuffer
5511                      <1>
5512                      <1> loc_move_mainprog_cfg_line:
5513 00009616 AC          <1>      lodsb
5514 00009617 3C20          <1>      cmp     al, 20h
5515 00009619 720C          <1>      jb      short loc_move_mainprog_cfg_n12
5516 0000961B AA          <1>      stosb
5517 0000961C E2F8          <1>      loop   loc_move_mainprog_cfg_line
5518                      <1>
5519                      <1> loc_move_mainprog_cfg_n11:
5520 0000961E 39D6          <1>      cmp     esi, edx ; + configuration file size
5521 00009620 7312          <1>      jnb     short loc_end_of_mainprog_cfg_line
5522 00009622 AC          <1>      lodsb
5523 00009623 3C20          <1>      cmp     al, 20h
5524 00009625 73F7          <1>      jnb     short loc_move_mainprog_cfg_n11
5525                      <1>
5526                      <1> loc_move_mainprog_cfg_n12:
5527 00009627 39D6          <1>      cmp     esi, edx
5528 00009629 7309          <1>      jnb     short loc_end_of_mainprog_cfg_line
5529 0000962B 8A06          <1>      mov     al, [esi]
5530 0000962D 3C20          <1>      cmp     al, 20h
5531 0000962F 7703          <1>      ja      short loc_end_of_mainprog_cfg_line
5532 00009631 46          <1>      inc     esi
5533 00009632 EBF3          <1>      jmp     short loc_move_mainprog_cfg_n12
5534                      <1>
5535                      <1> loc_end_of_mainprog_cfg_line:
5536 00009634 C60700      <1>      mov     byte [edi], 0
5537                      <1>
5538 00009637 8935[D8380100] <1>      mov     [MainProgCfg_LineOffset], esi
5539                      <1>
5540                      <1> loc_move_mainprog_cfg_command:
5541 0000963D BE[96390100] <1>      mov     esi, CommandBuffer
5542 00009642 89F7          <1>      mov     edi, esi
5543 00009644 31DB          <1>      xor     ebx, ebx
5544                      <1>      ;xor     ecx, ecx
5545 00009646 30C9          <1>      xor     cl, cl
5546                      <1>
5547                      <1> loc_move_mcfg_first_cmd_char:
5548 00009648 8A041E          <1>      mov     al, [esi+ebx]
5549 0000964B FEC3          <1>      inc     bl
5550 0000964D 3C20          <1>      cmp     al, 20h
5551 0000964F 7712          <1>      ja      short loc_move_mcfg_cmd_capitalizing
5552 00009651 7237          <1>      jb      short loc_move_mcfg_cmd_arguments_ok
5553 00009653 80FB4F          <1>      cmp     bl, 79
5554 00009656 72F0          <1>      jb      short loc_move_mcfg_first_cmd_char
5555 00009658 EB30          <1>      jmp     short loc_move_mcfg_cmd_arguments_ok
5556                      <1>
5557                      <1> loc_move_mcfg_next_cmd_char:

```

```

5558 0000965A 8A041E      <1>      mov     al, [esi+ebx]
5559 0000965D FEC3      <1>      inc     bl
5560 0000965F 3C20      <1>      cmp     al, 20h
5561 00009661 7614      <1>      jna     short loc_move_mcfg_cmd_ok
5562                                <1>
5563                                <1> loc_move_mcfg_cmd_capitalizing:
5564 00009663 3C61      <1>      cmp     al, 61h ; 'a'
5565 00009665 7206      <1>      jnb     short loc_move_mcfg_cmd_caps_ok
5566 00009667 3C7A      <1>      cmp     al, 7Ah ; 'z'
5567 00009669 7702      <1>      ja     short loc_move_mcfg_cmd_caps_ok
5568 0000966B 24DF      <1>      and     al, 0DFh ; sub     al, 'a'-'A'
5569                                <1>
5570                                <1> loc_move_mcfg_cmd_caps_ok:
5571 0000966D AA      <1>      stosb
5572 0000966E FEC1      <1>      inc     cl
5573 00009670 80FB4F      <1>      cmp     bl, 79
5574 00009673 72E5      <1>      jnb     short loc_move_mcfg_next_cmd_char
5575 00009675 EB13      <1>      jmp     short loc_move_mcfg_cmd_arguments_ok
5576                                <1>
5577                                <1> loc_move_mcfg_cmd_ok:
5578 00009677 30C0      <1>      xor     al, al ; 0
5579                                <1>
5580                                <1> loc_move_mcfg_cmd_arguments:
5581 00009679 8807      <1>      mov     [edi], al
5582 0000967B 47      <1>      inc     edi
5583 0000967C 80FB4F      <1>      cmp     bl, 79
5584 0000967F 7309      <1>      jnb     short loc_move_mcfg_cmd_arguments_ok
5585 00009681 8A041E      <1>      mov     al, [esi+ebx]
5586 00009684 FEC3      <1>      inc     bl
5587 00009686 3C20      <1>      cmp     al, 20h
5588 00009688 73EF      <1>      jnb     short loc_move_mcfg_cmd_arguments
5589                                <1>
5590                                <1> loc_move_mcfg_cmd_arguments_ok:
5591 0000968A C60700      <1>      mov     byte [edi], 0
5592                                <1>
5593                                <1> loc_mcfg_process_cmd_interpreter:
5594 0000968D E8F4DFFFFF      <1>      call    command_interpreter
5595                                <1>
5596                                <1> loc_mcfg_ci_return_addr:
5597 00009692 A1[D4380100]      <1>      mov     eax, [MainProgCfg_FileSize]
5598 00009697 89C2      <1>      mov     edx, eax
5599 00009699 8B35[D8380100]      <1>      mov     esi, [MainProgCfg_LineOffset]
5600 0000969F 01F2      <1>      add     edx, esi
5601 000096A1 0305[00450100]      <1>      add     eax, [csftdf_sf_mem_addr]
5602 000096A7 29F0      <1>      sub     eax, esi
5603 000096A9 0F874EFFFFFF      <1>      ja     loc_mcfg_process_next_line_check
5604                                <1>
5605 000096AF E81D000000      <1>      call    mcfg_deallocate_mem
5606                                <1>
5607 000096B4 B94F000000      <1>      mov     ecx, 79 ; 80 ?
5608 000096B9 BF[96390100]      <1>      mov     edi, CommandBuffer
5609 000096BE 30C0      <1>      xor     al, al
5610 000096C0 F3AA      <1>      rep     stosb
5611                                <1>
5612                                <1> ; 06/05/2016
5613 000096C2 BE[13070100]      <1>      mov     esi, nextline
5614 000096C7 E8F1CBFFFF      <1>      call    print_msg
5615 000096CC E92DD6FFFF      <1>      jmp     dos_prompt
5616                                <1>
5617                                <1> mcfg_deallocate_mem:
5618 000096D1 A1[00450100]      <1>      mov     eax, [csftdf_sf_mem_addr] ; start address
5619 000096D6 8B0D[04450100]      <1>      mov     ecx, [csftdf_sf_mem_bsize] ; block size
5620                                <1> ;call deallocate_memory_block
5621                                <1> ;retn
5622 000096DC E943BFFFFF      <1>      jmp     deallocate_memory_block
5623                                <1>
5624                                <1> mcfg_read_file_sectors:
5625                                <1> ; 14/04/2016
5626 000096E1 807E0300      <1>      cmp     byte [esi+LD_FATType], 0
5627 000096E5 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 000096E7 8B15[D4380100]      <1>      mov     edx, [MainProgCfg_FileSize]
5637 000096ED 2B15[18450100]      <1>      sub     edx, [csftdf_sf_rbytes]
5638 000096F3 3B15[10450100]      <1>      cmp     edx, [csftdf_r_size]
5639 000096F9 7306      <1>      jnb     short mcfg_read_fat_file_secs_1
5640 000096FB 8915[10450100]      <1>      mov     [csftdf_r_size], edx
5641                                <1>
5642                                <1> mcfg_read_fat_file_secs_1:
5643 00009701 A1[10450100]      <1>      mov     eax, [csftdf_r_size]
5644 00009706 29D2      <1>      sub     edx, edx
5645 00009708 0FB74E11      <1>      movzx   ecx, word [esi+LD_BPB+BytesPerSec]
5646 0000970C 01C8      <1>      add     eax, ecx
5647 0000970E 48      <1>      dec     eax
5648 0000970F F7F1      <1>      div     ecx
5649 00009711 89C1      <1>      mov     ecx, eax ; sector count
5650 00009713 A1[08450100]      <1>      mov     eax, [csftdf_sf_cluster]
5651                                <1>
5652                                <1> ; EBX = memory block address (current)
5653                                <1>
5654 00009718 E88E230000      <1>      call    read_fat_file_sectors
5655 0000971D 7230      <1>      jc     short mcfg_read_fat_file_secs_3
5656                                <1>
5657                                <1> ; EBX = next memory address
5658                                <1>
5659 0000971F A1[18450100]      <1>      mov     eax, [csftdf_sf_rbytes]

```

```

5660 00009724 0305[10450100] <1> add    eax, [csftdf_r_size]
5661 0000972A 8B15[D4380100] <1> mov    edx, [MainProgCfg_FileSize]
5662 00009730 39D0 <1> cmp    eax, edx
5663 00009732 731B <1> jnb    short mcfg_read_fat_file_secs_3 ; edx > 0
5664 00009734 A3[18450100] <1> mov    [csftdf_sf_rbytes], eax
5665 <1>
5666 00009739 53 <1> push   ebx ; *
5667 <1> ; get next cluster (csftdf_r_size! bytes)
5668 0000973A A1[08450100] <1> mov    eax, [csftdf_sf_cluster]
5669 0000973F E839210000 <1> call   get_next_cluster
5670 00009744 5B <1> pop    ebx ; *
5671 00009745 7301 <1> jnc    short mcfg_read_fat_file_secs_2
5672 <1>
5673 <1> ;mov    eax, 17; Read error !
5674 00009747 C3 <1> retn
5675 <1>
5676 <1> mcfg_read_fat_file_secs_2:
5677 00009748 29D2 <1> sub    edx, edx ; 0
5678 0000974A A3[08450100] <1> mov    [csftdf_sf_cluster], eax ; next cluster
5679 <1>
5680 <1> mcfg_read_fat_file_secs_3:
5681 0000974F C3 <1> retn
5682 <1>
5683 <1> mcfg_read_fs_file_sectors:
5684 00009750 C3 <1> retn
5685 <1>
5686 <1> loc_mcfg_load_fs_file:
5687 00009751 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 00009752 803E20 <1> cmp    byte [esi], 20h
5708 00009755 0F82F2E2FFFF <1> jb     loc_cmd_failed
5709 0000975B 7703 <1> ja     short loc_run_check_filename_ok
5710 0000975D 46 <1> inc    esi
5711 0000975E EBF2 <1> jmp    short loc_run_check_filename
5712 <1>
5713 <1> loc_run_check_filename_ok:
5714 00009760 C605[47390100]00 <1> mov    byte [CmdArgStart], 0 ; reset
5715 00009767 56 <1> push   esi ; *
5716 <1> loc_run_get_first_arg_pos:
5717 00009768 46 <1> inc    esi
5718 00009769 8A06 <1> mov    al, [esi]
5719 0000976B 3C20 <1> cmp    al, 20h
5720 0000976D 77F9 <1> ja     short loc_run_get_first_arg_pos
5721 0000976F C60600 <1> mov    byte [esi], 0
5722 <1> loc_run_get_external_arg_pos:
5723 <1> ; 11/05/2016
5724 00009772 46 <1> inc    esi
5725 00009773 8A06 <1> mov    al, [esi]
5726 00009775 3C20 <1> cmp    al, 20h
5727 00009777 760C <1> jna    short loc_run_parse_path_name
5728 00009779 89F0 <1> mov    eax, esi
5729 0000977B 2D[96390100] <1> sub    eax, CommandBuffer
5730 00009780 A2[47390100] <1> mov    byte [CmdArgStart], al
5731 <1> loc_run_parse_path_name:
5732 00009785 5E <1> pop    esi ; *
5733 00009786 BF[8A420100] <1> mov    edi, FindFile_Drv
5734 0000978B E8D6090000 <1> call   parse_path_name
5735 00009790 0F82B7E2FFFF <1> jc     loc_cmd_failed
5736 <1>
5737 <1> loc_run_check_filename_exists:
5738 00009796 BE[CC420100] <1> mov    esi, FindFile_Name
5739 0000979B 803E20 <1> cmp    byte [esi], 20h
5740 0000979E 0F86A9E2FFFF <1> jna    loc_cmd_failed
5741 <1>
5742 <1> loc_run_check_exe_filename_ext:
5743 000097A4 E891020000 <1> call   check_prg_filename_ext
5744 000097A9 0F829EE2FFFF <1> jc     loc_cmd_failed
5745 <1>
5746 <1> loc_run_check_exe_filename_ext_ok:
5747 000097AF 66A3[A2450100] <1> mov    word [EXE_ID], ax
5748 <1>
5749 <1> loc_run_drv:
5750 000097B5 C605[A1450100]00 <1> mov    byte [Run_Manual_Path], 0
5751 000097BC A1[E0380100] <1> mov    eax, [Current_Dir_FCluster]
5752 000097C1 A3[9C450100] <1> mov    [Run_CDirFC], eax
5753 <1> ;
5754 000097C6 8A35[E6380100] <1> mov    dh, [Current_Drv]
5755 000097CC 8835[46410100] <1> mov    [RUN_CDRV], dh
5756 <1>
5757 000097D2 8A15[8A420100] <1> mov    dl, [FindFile_Drv]
5758 000097D8 38F2 <1> cmp    dl, dh
5759 000097DA 7412 <1> je     short loc_run_change_directory
5760 <1>
5761 000097DC 8005[A1450100]02 <1> add    byte [Run_Manual_Path], 2

```


5762		<1>	
5763	000097E3 E8D5D3FFFF	<1>	call change_current_drive
5764	000097E8 0F828AE2FFFF	<1>	jc loc_run_cmd_failed
5765		<1>	
5766		<1>	loc_run_change_directory:
5767	000097EE 803D[8B420100]20	<1>	cmp byte [FindFile_Directory], 20h
5768	000097F5 7623	<1>	jna short loc_run_find_executable_file
5769		<1>	
5770	000097F7 FE05[A1450100]	<1>	inc byte [Run_Manual_Path]
5771		<1>	
5772	000097FD FE05[A8FA0000]	<1>	inc byte [Restore_CDIRE]
5773		<1>	
5774	00009803 BE[8B420100]	<1>	mov esi, FindFile_Directory
5775	00009808 30E4	<1>	xor ah, ah ; CD_COMMAND sign -> 0
5776	0000980A E843030000	<1>	call change_current_directory
5777	0000980F 0F8263E2FFFF	<1>	jc loc_run_cmd_failed
5778		<1>	
5779		<1>	loc_run_change_prompt_dir_string:
5780	00009815 E858020000	<1>	call change_prompt_dir_string
5781		<1>	
5782		<1>	loc_run_find_executable_file:
5783	0000981A 66C705[A0450100]00-	<1>	mov word [Run_Auto_Path], 0
5783	00009822 00	<1>	
5784		<1>	
5785		<1>	loc_run_find_executable_file_next:
5786	00009823 BE[CC420100]	<1>	mov esi, FindFile_Name
5787		<1>	loc_run_find_program_file_next:
5788	00009828 66B80018	<1>	mov ax, 1800h ; Except volume label and dirs
5789	0000982C 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	00009831 0F835C010000	<1>	jnc loc_load_and_run_file
5794		<1>	
5795	00009837 3C02	<1>	cmp al, 2 ; file not found
5796	00009839 0F8539E2FFFF	<1>	jne loc_run_cmd_failed
5797		<1>	
5798	0000983F 66A1[A2450100]	<1>	mov ax, word [EXE_ID]
5799	00009845 80FC2E	<1>	cmp ah, '.' ; File name has extension sign
5800	00009848 7424	<1>	je short loc_run_check_auto_path
5801		<1>	
5802	0000984A 08C0	<1>	or al, al
5803	0000984C 7520	<1>	jnz short loc_run_check_auto_path
5804		<1>	
5805	0000984E 80FC08	<1>	cmp ah, 8 ; count of file name chars
5806	00009851 771B	<1>	ja short loc_run_check_auto_path
5807		<1>	
5808		<1>	loc_run_change_file_ext_to_prg:
5809	00009853 0FB6DC	<1>	movzx ebx, ah ; count of file name chars
5810	00009856 BE[CC420100]	<1>	mov esi, FindFile_Name
5811	0000985B 01F3	<1>	add ebx, esi
5812		<1>	; 07/05/2016
5813	0000985D C7032E505247	<1>	mov dword [ebx], '.PRG'
5814	00009863 66C705[A2450100]50-	<1>	mov word [EXE_ID], 'P.'
5814	0000986B 2E	<1>	
5815	0000986C 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	0000986E A0[A1450100]	<1>	mov al, [Run_Manual_Path]
5826	00009873 08C0	<1>	or al, al
5827	00009875 0F85D2E1FFFF	<1>	jnz loc_cmd_failed
5828		<1>	
5829		<1>	loc_run_check_auto_path_again:
5830	0000987B 66833D[A0450100]FF	<1>	cmp word [Run_Auto_Path], 0FFFFh
5831		<1>	; 0FFFFh = Not a valid run path (in ENV block)
5832	00009883 0F83C4E1FFFF	<1>	jnb loc_cmd_failed
5833		<1>	; xor al, al
5834	00009889 BE[74FB0000]	<1>	mov esi, Cmd_Path ; 'PATH'
5835	0000988E BF[E6390100]	<1>	mov edi, TextBuffer
5836	00009893 E857F9FFFF	<1>	call get_environment_string
5837	00009898 730E	<1>	jnc short loc_run_chk_filename_ext_again
5838	0000989A 66C705[A0450100]FF-	<1>	mov word [Run_Auto_Path], 0FFFFh ; invalid
5838	000098A2 FF	<1>	
5839	000098A3 E9A5E1FFFF	<1>	jmp loc_cmd_failed
5840		<1>	
5841		<1>	loc_run_chk_filename_ext_again:
5842	000098A8 89C1	<1>	mov ecx, eax ; string length (with zero tail)
5843	000098AA 49	<1>	dec ecx ; without zero tail
5844	000098AB 66A1[A2450100]	<1>	mov ax, [EXE_ID]
5845	000098B1		

```

5861 000098D2 6609C0      <1>      or      ax, ax
5862 000098D5 7502      <1>      jnz     short loc_run_auto_path_pos_move
5863 000098D7 B005      <1>      mov     al, 5
5864                      <1>
5865                      <1> loc_run_auto_path_pos_move:
5866 000098D9 89FE      <1>      mov     esi, edi ; offset TextBuffer
5867 000098DB 01C6      <1>      add     esi, eax
5868                      <1>
5869                      <1> loc_run_auto_path_pos_space_loop:
5870 000098DD AC        <1>      lodsb
5871 000098DE 3C20      <1>      cmp     al, 20h
5872 000098E0 74FB      <1>      je      short loc_run_auto_path_pos_space_loop
5873 000098E2 0F8265E1FFFF <1>      jb      loc_cmd_failed
5874 000098E8 AA        <1>      stosb
5875                      <1> loc_run_auto_path_pos_move_next:
5876 000098E9 AC        <1>      lodsb
5877 000098EA 3C3B      <1>      cmp     al, ';'
5878 000098EC 7414      <1>      je      short loc_run_auto_path_pos_move_last_byte
5879 000098EE 3C20      <1>      cmp     al, 20h
5880 000098F0 74F7      <1>      je      short loc_run_auto_path_pos_move_next
5881 000098F2 7203      <1>      jnb     short loc_byte_ptr_end_of_path
5882 000098F4 AA        <1>      stosb
5883 000098F5 EBF2      <1>      jmp     short loc_run_auto_path_pos_move_next
5884                      <1>
5885                      <1> loc_byte_ptr_end_of_path:
5886 000098F7 66C705[A0450100]FF- <1>      mov     word [Run_Auto_Path], 0FFFFh ; end of path
5886 000098FF FF        <1>
5887 00009900 EB0D      <1>      jmp     short loc_run_auto_path_move_ok
5888                      <1>
5889                      <1> loc_run_auto_path_pos_move_last_byte:
5890 00009902 89F0      <1>      mov     eax, esi
5891 00009904 2D[E6390100]    <1>      sub     eax, TextBuffer
5892 00009909 66A3[A0450100]    <1>      mov     [Run_Auto_Path], ax ; next path position
5893                      <1>
5894                      <1> loc_run_auto_path_move_ok:
5895 0000990F 4F        <1>      dec     edi
5896 00009910 B02F      <1>      mov     al, '/'
5897 00009912 3807      <1>      cmp     [edi], al
5898 00009914 7403      <1>      je      short loc_run_auto_path_move_file_name
5899 00009916 47        <1>      inc     edi
5900 00009917 8807      <1>      mov     [edi], al
5901                      <1>
5902                      <1> loc_run_auto_path_move_file_name:
5903 00009919 47        <1>      inc     edi
5904 0000991A BE[CC420100] <1>      mov     esi, FindFile_Name
5905                      <1>
5906                      <1> loc_run_auto_path_move_fn_loop:
5907 0000991F AC        <1>      lodsb
5908 00009920 AA        <1>      stosb
5909 00009921 08C0      <1>      or      al, al
5910 00009923 75FA      <1>      jnz     short loc_run_auto_path_move_fn_loop
5911                      <1>
5912 00009925 BE[E6390100] <1>      mov     esi, TextBuffer
5913 0000992A BF[8A420100] <1>      mov     edi, FindFile_Drv
5914 0000992F E832080000 <1>      call    parse_path_name
5915 00009934 0F8213E1FFFF <1>      jc      loc_cmd_failed
5916                      <1>
5917 0000993A 8A35[E6380100]    <1>      mov     dh, [Current_Drv]
5918 00009940 8A15[8A420100]    <1>      mov     dl, [FindFile_Drv]
5919 00009946 38F2      <1>      cmp     dl, dh
5920 00009948 740B      <1>      je      short loc_run_change_directory_again
5921                      <1>
5922 0000994A E86ED2FFFF <1>      call    change_current_drive
5923 0000994F 0F8223E1FFFF <1>      jc      loc_run_cmd_failed
5924                      <1>
5925                      <1> loc_run_change_directory_again:
5926 00009955 803D[8B420100]20 <1>      cmp     byte [FindFile_Directory], 20h
5927 0000995C 761D      <1>      jna     short loc_load_executable_cdir_chk_again
5928                      <1>
5929 0000995E FE05[A8FA0000]    <1>      inc     byte [Restore_CDIRE]
5930 00009964 BE[8B420100] <1>      mov     esi, FindFile_Directory
5931 00009969 30E4      <1>      xor     ah, ah ; CD_COMMAND sign -> 0
5932 0000996B E8E2010000 <1>      call    change_current_directory
5933 00009970 0F8202E1FFFF <1>      jc      loc_run_cmd_failed
5934                      <1>
5935                      <1> loc_run_chg_prompt_dir_str_again:
5936 00009976 E8F7000000 <1>      call    change_prompt_dir_string
5937                      <1>
5938                      <1> loc_load_executable_cdir_chk_again:
5939 0000997B A1[E0380100]    <1>      mov     eax, [Current_Dir_FCluster]
5940 00009980 3B05[9C450100]    <1>      cmp     eax, [Run_CDireFC]
5941 00009986 0F8597FEFFFF <1>      jne     loc_run_find_executable_file_next
5942 0000998C 30C0      <1>      xor     al, al ; 0
5943 0000998E 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 00009993 BE[CC420100] <1>      mov     esi, FindFile_Name
5949 00009998 BF[E6390100] <1>      mov     edi, TextBuffer
5950                      <1>
5951                      <1>      ; 24/04/2016
5952 0000999D 31D2      <1>      xor     edx, edx
5953 0000999F 668915[924F0100] <1>      mov     word [argc], dx ; 0
5954 000099A6 8915[D44E0100]    <1>      mov     dword [u.nread], edx ; 0
5955                      <1>
5956                      <1> loc_load_and_run_file_1:
5957 000099AC AC        <1>      lodsb
5958 000099AD AA        <1>      stosb
5959 000099AE FF05[D44E0100] <1>      inc     dword [u.nread]
5960 000099B4 20C0      <1>      and     al, al
5961 000099B6 75F4      <1>      jnz     short loc_load_and_run_file_1

```

```

5962                                     <1>
5963 000099B8 A0[47390100]             <1>      mov     al, [CmdArgStart]
5964 000099BD 20C0                     <1>      and     al, al
5965 000099BF 7445                     <1>      jz      short loc_load_and_run_file_7
5966                                     <1>
5967 000099C1 0FB6F0                     <1>      movzx   esi, al ; 11/05/2016
5968 000099C4 B950000000               <1>      mov     ecx, 80
5969 000099C9 29F1                     <1>      sub     ecx, esi
5970 000099CB 81C6[96390100]           <1>      add     esi, CommandBuffer
5971                                     <1>
5972 000099D1 66FF05[924F0100]         <1>      inc     word [argc] ; 11/05/2016
5973                                     <1>
5974                                     <1> loc_load_and_run_file_2:
5975 000099D8 AC                         <1>      lodsb
5976 000099D9 3C20                     <1>      cmp     al, 20h
5977 000099DB 7717                     <1>      ja      short loc_load_and_run_file_5
5978 000099DD 721E                     <1>      jb      short loc_load_and_run_file_6
5979                                     <1>
5980                                     <1> loc_load_and_run_file_3:
5981 000099DF 803E20                   <1>      cmp     byte [esi], 20h
5982 000099E2 7707                     <1>      ja      short loc_load_and_run_file_4
5983 000099E4 7217                     <1>      jb      short loc_load_and_run_file_6
5984 000099E6 46                       <1>      inc     esi
5985 000099E7 E2F6                     <1>      loop    loc_load_and_run_file_3
5986 000099E9 EB12                     <1>      jmp     short loc_load_and_run_file_6
5987                                     <1>
5988                                     <1> loc_load_and_run_file_4:
5989 000099EB 28C0                     <1>      sub     al, al ; 0
5990 000099ED 66FF05[924F0100]         <1>      inc     word [argc]
5991                                     <1> loc_load_and_run_file_5:
5992 000099F4 AA                         <1>      stosb
5993 000099F5 FF05[D44E0100]           <1>      inc     dword [u.nread]
5994 000099FB E2DB                     <1>      loop    loc_load_and_run_file_2
5995                                     <1>
5996                                     <1> loc_load_and_run_file_6:
5997 000099FD 30C0                     <1>      xor     al, al ; 0
5998 000099FF AA                         <1>      stosb
5999 00009A00 FF05[D44E0100]           <1>      inc     dword [u.nread]
6000                                     <1> loc_load_and_run_file_7:
6001 00009A06 8807                     <1>      mov     [edi], al ; 0
6002 00009A08 66FF05[924F0100]         <1>      inc     word [argc] ; 24/04/2016
6003 00009A0F FF05[D44E0100]           <1>      inc     dword [u.nread] ; 24/04/2016
6004 00009A15 BE[E6390100]             <1>      mov     esi, TextBuffer
6005 00009A1A 8B15[F8420100]           <1>      mov     edx, [FindFile_DirEntry+DirEntry_FileSize]
6006 00009A20 66A1[F0420100]           <1>      mov     ax, [FindFile_DirEntry+DirEntry_FstClusHI]
6007 00009A26 66C1E010                <1>      shl     ax, 16
6008 00009A2A 66A1[F6420100]           <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 00009A30 E898450000               <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 00009A35 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 00009A3A 30E4                     <1>      xor     ah, ah
6037                                     <1> loc_run_check_filename_ext:
6038 00009A3C AC                         <1>      lodsb
6039 00009A3D 3C21                     <1>      cmp     al, 21h
6040 00009A3F 7229                     <1>      jb      short loc_check_exe_fn_retn
6041 00009A41 FEC4                     <1>      inc     ah
6042 00009A43 3C2E                     <1>      cmp     al, '.'
6043 00009A45 75F5                     <1>      jne     short loc_run_check_filename_ext
6044                                     <1>
6045                                     <1> loc_run_check_filename_ext_dot:
6046 00009A47 80FC02                   <1>      cmp     ah, 2 ; .??? is not valid
6047 00009A4A 88C4                     <1>      mov     ah, al ; '.'
6048 00009A4C 7219                     <1>      jb      short loc_check_prg_fn_retn
6049                                     <1>
6050                                     <1> loc_run_check_filename_ext_dot_ok:
6051 00009A4E AC                         <1>      lodsb
6052 00009A4F 24DF                     <1>      and     al, 0DFh
6053                                     <1>
6054                                     <1> loc_run_check_filename_ext_prg:
6055 00009A51 3C50                     <1>      cmp     al, 'P'
6056 00009A53 7212                     <1>      jb      short loc_check_prg_fn_retn
6057 00009A55 7711                     <1>      ja      short loc_check_prg_fn_stc
6058 00009A57 AC                         <1>      lodsb
6059 00009A58 24DF                     <1>      and     al, 0DFh
6060 00009A5A 3C52                     <1>      cmp     al, 'R'
6061 00009A5C 750A                     <1>      jne     short loc_check_prg_fn_stc
6062 00009A5E AC                         <1>      lodsb
6063 00009A5F 24DF                     <1>      and     al, 0DFh

```

```

6064 00009A61 3C47      <1>      cmp     al, 'G'
6065 00009A63 7503      <1>      jne     short loc_check_prg_fn_stc
6066                                <1>
6067 00009A65 B050      <1>      mov     al, 'P'
6068                                <1> loc_check_prg_fn_retn:
6069 00009A67 C3          <1>      retn
6070                                <1>
6071                                <1> loc_check_prg_fn_stc:
6072 00009A68 F9          <1>      stc
6073 00009A69 C3          <1>      retn
6074                                <1>
6075                                <1> loc_check_exe_fn_retn:
6076 00009A6A 28C0      <1>      sub     al, al ; 0
6077 00009A6C C3          <1>      retn
6078                                <1>
6079                                <1> find_and_list_files:
6080 00009A6D C3          <1>      retn
6081                                <1> set_exec_arguments:
6082 00009A6E C3          <1>      retn
6083                                <1> delete_fs_directory:
6084 00009A6F 31C0      <1>      xor     eax, eax
6085 00009A71 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
26                               <1>
27 00009A72 BE[47410100] <1>      mov     esi, PATH_Array
28                                <1> change_prompt_dir_str: ; 05/10/2016 (call from 'set_working_path')
29 00009A77 BF[EA380100] <1>      mov     edi, Current_Directory
30 00009A7C 8A25[E4380100] <1>      mov     ah, [Current_Dir_Level]
31 00009A82 E807000000 <1>      call    set_current_directory_string
32 00009A87 880D[45390100] <1>      mov     [Current_Dir_StrLen], cl
33                                <1>
34 00009A8D C3          <1>      retn
35                                <1>
36                                <1> set_current_directory_string:
37                                <1>      ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)
38                                <1>      ; 27/03/2011
39                                <1>      ; 09/10/2009
40                                <1>      ; INPUT:
41                                <1>      ;     ESI = Path Array Address
42                                <1>      ;     EDI = Current Directory String Buffer
43                                <1>      ;     AH = Current Directory Level
44                                <1>      ; OUTPUT => EAX, EBX, ESI will be changed
45                                <1>      ;     EDI will be same with input
46                                <1>      ;     ECX = Current Directory String Length
47                                <1>
48 00009A8E 57          <1>      push     edi
49 00009A8F 80FC00      <1>      cmp     ah, 0
50 00009A92 7652      <1>      jna     short pass_write_path
51 00009A94 83C610      <1>      add     esi, 16
52 00009A97 89F3      <1>      mov     ebx, esi
53                                <1> loc_write_path:
54 00009A99 B908000000 <1>      mov     ecx, 8
55                                <1> path_write_dirname1:
56 00009A9E AC          <1>      lodsb
57 00009A9F 3C20      <1>      cmp     al, 20h
58 00009AA1 7612      <1>      jna     short pass_write_dirname1
59 00009AA3 AA          <1>      stosb
60 00009AA4 81FF[44390100] <1>      cmp     edi, End_Of_Current_Dir_Str
61 00009AAA 733A      <1>      jnb     short pass_write_path
62 00009AAC E2F0      <1>      loop    path_write_dirname1
63 00009AAE 803E20      <1>      cmp     byte [esi], 20h
64 00009AB1 7624      <1>      jna     short pass_write_dirname2
65 00009AB3 EB0A      <1>      jmp     short loc_put_dot_cont_ext
66                                <1> pass_write_dirname1:
67 00009AB5 89DE      <1>      mov     esi, ebx
68 00009AB7 83C608      <1>      add     esi, 8
69 00009ABA 803E20      <1>      cmp     byte [esi], 20h
70 00009ABD 7618      <1>      jna     short pass_write_dirname2
71                                <1> loc_put_dot_cont_ext:
72 00009ABF C6072E      <1>      mov     byte [edi], "."
73                                <1>      ;mov     ecx, 3
74 00009AC2 B103      <1>      mov     cl, 3
75                                <1> loc_check_dir_name_ext:
76 00009AC4 AC          <1>      lodsb
77 00009AC5 47          <1>      inc     edi
78 00009AC6 3C20      <1>      cmp     al, 20h
79 00009AC8 760D      <1>      jna     short pass_write_dirname2

```


80	00009ACA	8807	<1>	mov	[edi], al
81	00009ACC	81FF[44390100]	<1>	cmp	edi, End_Of_Current_Dir_Str
82	00009AD2	7312	<1>	jnb	short pass_write_path
83	00009AD4	E2EE	<1>	loop	loc_check_dir_name_ext
84	00009AD6	47	<1>	inc	edi
85			<1>	pass_write_dirname2:	
86	00009AD7	FECC	<1>	dec	ah
87	00009AD9	740B	<1>	jz	short pass_write_path
88	00009ADB	83C310	<1>	add	ebx, 16
89	00009ADE	89DE	<1>	mov	esi, ebx
90	00009AE0	C6072F	<1>	mov	byte [edi], "/"
91	00009AE3	47	<1>	inc	edi
92	00009AE4	EBB3	<1>	jmp	short loc_write_path
93			<1>	pass_write_path:	
94	00009AE6	C60700	<1>	mov	byte [edi], 0
95	00009AE9	47	<1>	inc	edi
96	00009AEA	89F9	<1>	mov	ecx, edi
97	00009AEC	5F	<1>	pop	edi
98	00009AED	29F9	<1>	sub	ecx, edi
99			<1>		; ECX = Current Directory String Length
100	00009AEF	C3	<1>	retn	
101			<1>		
102			<1>	get_current_directory:	
103			<1>		; 15/10/2016
104			<1>		; 14/02/2016
105			<1>		; 24/01/2016 (TRDOS 386 = TRDOS v2.0)
106			<1>		; 27/03/2011
107			<1>		;
108			<1>		; INPUT-> ESI = Current Directory Buffer
109			<1>		; DL = TRDOS Logical Dos Drive Number + 1
110			<1>		; (0= Default/Current Drive)
111			<1>		;
112			<1>		; Note: Required dir buffer length may be <= 92 bytes
113			<1>		; for TRDOS (7*12 name chars + 7 slash + 0)
114			<1>		; OUTPUT -> ESI = Current Directory Buffer
115			<1>		; EAX, EBX, ECX, EDX, EDI will be changed
116			<1>		; CX/CL = Current Directory String Length
117			<1>		; DL = Drive Number (0 based)
118			<1>		; (If input is 0, output is current drv number)
119			<1>		; DH = same with input
120			<1>		; cf = 0 -> AL = 0
121			<1>		; cf = 1 -> error code in AL
122			<1>		
123			<1>	loc_get_current_drive_0:	
124	00009AF0	80FA00	<1>	cmp	dl, 0
125	00009AF3	7708	<1>	ja	short loc_get_current_drive_1
126	00009AF5	8A15[E6380100]	<1>	mov	dl, [Current_Drv]
127	00009AFB	EB17	<1>	jmp	short loc_get_current_drive_2
128			<1>	loc_get_current_drive_1:	
129	00009AFD	FECA	<1>	dec	dl
130	00009AFF	3A15[A7FA0000]	<1>	cmp	dl, [Last_DOS_DiskNo]
131	00009B05	760D	<1>	jna	short loc_get_current_drive_2
132	00009B07	B80F000000	<1>	mov	eax, 0Fh ; Invalid drive (Drive not ready!)
133	00009B0C	F5	<1>	cmc	; stc
134	00009B0D	C3	<1>	retn	
135			<1>		
136			<1>	loc_get_current_drive_not_ready_retn:	
137	00009B0E	5E	<1>	pop	esi
138			<1>	;mov	eax, 15
139	00009B0F	66B80F00	<1>	mov	ax, 15 ; Drive not ready
140	00009B13	C3	<1>	retn	
141			<1>		
142			<1>	loc_get_current_drive_2:	
143	00009B14	31C0	<1>	xor	eax, eax
144	00009B16	88D4	<1>	mov	ah, dl
145	00009B18	56	<1>	push	esi
146	00009B19	BE00010900	<1>	mov	esi, Logical_DOSDisks
147	00009B1E	01C6	<1>	add	esi, eax
148	00009B20	8A06	<1>	mov	al, [esi+LD_Name]
149	00009B22	3C41	<1>	cmp	al, 'A'
150	00009B24	72E8	<1>	jnb	short loc_get_current_drive_not_ready_retn
151			<1>		
152	00009B26	8A667F	<1>	mov	ah, [esi+LD_CDirLevel]
153	00009B29	08E4	<1>	or	ah, ah
154	00009B2B	7506	<1>	jnz	short loc_get_current_drive_3
155			<1>		
156			<1>	;xor	ah, ah ; mov ah, 0
157	00				

```

182      <1>      ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
183      <1>      ; 18/09/2011 (DIR.ASM, 09/10/2011)
184      <1>      ; 04/10/2009
185      <1>      ; 2005
186      <1>      ; INPUT ->
187      <1>      ;      ESI = Directory string
188      <1>      ;      ah = CD command (CDh = save current dir string)
189      <1>      ; OUTPUT ->
190      <1>      ;      EDI = DOS Drive Description Table
191      <1>      ;      cf = 1 -> error
192      <1>      ;      EAX = Error code
193      <1>      ;      cf = 0 -> succesful
194      <1>      ;      ESI = PATH_Array
195      <1>      ;      EAX = Current Directory First Cluster
196      <1>      ;
197      <1>      ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
198      <1>
199 00009B52 8825[D5410100] <1>      mov     [CD_COMMAND], ah
200 00009B58 803E2F <1>      cmp     byte [esi], '/'
201 00009B5B 7505 <1>      jne     short loc_ccd_cdir_level
202 00009B5D 46 <1>      inc     esi
203 00009B5E 30C0 <1>      xor     al, al
204 00009B60 EB05 <1>      jmp     short loc_ccd_parse_path_name
205 <1> loc_ccd_cdir_level:
206 00009B62 A0[E4380100] <1>      mov     al, [Current_Dir_Level]
207 <1> loc_ccd_parse_path_name:
208 00009B67 88C4 <1>      mov     ah, al
209 00009B69 BF[47410100] <1>      mov     edi, PATH_Array
210 <1>
211 <1> ; Reset directory levels > cdir level
212 <1> ; is this required !?
213 <1> ;
214 <1> ; Relations:
215 <1> ; MAINPROG.ASM (pass_ccdrv_reset_cdir_FAT_fcluster)
216 <1> ; proc_parse_dir_name,
217 <1> ; proc_change_current_directory (this procedure)
218 <1> ; proc_change_prompt_dir_string
219 <1>
220 00009B6E 0FB6C8 <1>      movzx   ecx, al
221 00009B71 FEC1 <1>      inc     cl
222 00009B73 C0E104 <1>      shl     cl, 4
223 00009B76 01CF <1>      add     edi, ecx
224 00009B78 B107 <1>      mov     cl, 7
225 00009B7A 28C1 <1>      sub     cl, al
226 00009B7C C0E102 <1>      shl     cl, 2
227 00009B7F 89C3 <1>      mov     ebx, eax
228 00009B81 31C0 <1>      xor     eax, eax ; 0
229 00009B83 F3AB <1>      rep     stosd
230 00009B85 89D8 <1>      mov     eax, ebx
231 <1>
232 00009B87 BF[47410100] <1>      mov     edi, PATH_Array
233 <1>
234 00009B8C 803E20 <1>      cmp     byte [esi], 20h
235 00009B8F F5 <1>      cmc
236 00009B90 7305 <1>      jnc     short pass_ccd_parse_dir_name
237 <1>
238 <1>      ; ESI = Path name
239 <1>      ; AL = CCD_Level
240 00009B92 E872010000 <1>      call    parse_dir_name
241 <1>      ; AL = CCD_Level
242 <1>      ; AH = Last_Dir_Level
243 <1>      ; (EDI = PATH_Array)
244 <1>
245 <1> pass_ccd_parse_dir_name:
246 00009B97 9C <1>      pushf
247 <1>
248 <1>      ;mov [CCD_Level], al
249 <1>      ;mov [Last_Dir_Level], ah
250 00009B98 66A3[CB410100] <1>      mov     [CCD_Level], ax
251 <1>
252 00009B9E 31DB <1>      xor     ebx, ebx
253 00009BA0 8A3D[E6380100] <1>      mov     bh, [Current_Drv]
254 00009BA6 BE00010900 <1>      mov     esi, Logical_DOSDisks
255 00009BAB 01DE <1>      add     esi, ebx
256 <1>
257 00009BAD 9D <1>      popf
258 00009BAE 720A <1>      jc     short loc_ccd_bad_path_name_retn
259 <1>
260 00009BB0 8935[C7410100] <1>      mov     [CCD_DriveDT], esi
261 <1>
262 00009BB6 3C07 <1>      cmp     al, 7
263 00009BB8 7209 <1>      jb     short loc_ccd_load_child_dir
264 <1>
265 <1> loc_ccd_bad_path_name_retn:
266 00009BBA 87F7 <1>      xchg    esi, edi
267 00009BBC B813000000 <1>      mov     eax, 19 ; Bad directory/path name
268 00009BC1 F9 <1>      stc
269 <1> loc_ccd_retn_p:
270 00009BC2 C3 <1>      retn
271 <1>
272 <1> loc_ccd_load_child_dir:
273 <1>      ; AL = CCD_Level
274 00009BC3 08C0 <1>      or     al, al
275 00009BC5 7468 <1>      jz     short loc_ccd_load_root_dir
276 <1>
277 00009BC7 6689C1 <1>      mov     cx, ax
278 00009BCA C0E004 <1>      shl     al, 4
279 00009BCD 0FB6F0 <1>      movzx   esi, al
280 00009BD0 01FE <1>      add     esi, edi ; offset PATH_Array
281 <1>
282 00009BD2 8B460C <1>      mov     eax, [esi+12]
283 00009BD5 38E9 <1>      cmp     cl, ch

```

```

284 00009BD7 0F84FA000000 <1> je loc_ccd_load_sub_directory
285 00009BDD A3[E0380100] <1> mov [Current_Dir_FCluster], eax
286 <1>
287 <1> loc_ccd_load_child_dir_next:
288 00009BE2 83C610 <1> add esi, 16 ; DOS DirEntry Format FileName Address
289 <1>
290 <1> ; Directory attribute : 10h
291 00009BE5 B010 <1> mov al, 00010000b ; 10h (Attrib AND mask)
292 <1> ;mov ah, 11001000b ; C8h
293 <1> ; Volume name attribute: 8h
294 00009BE7 B408 <1> mov ah, 00001000b ; 08h (Attrib NAND, AND --> zero mask)
295 <1>
296 00009BE9 6631C9 <1> xor cx, cx
297 00009BEC E8B5010000 <1> call locate_current_dir_file
298 00009BF1 7353 <1> jnc short loc_ccd_set_dir_cluster_ptr
299 <1>
300 <1> ; 19/02/2016
301 <1> ;mov edi, [CCD_DriveDT]
302 00009BF3 8A25[CB410100] <1> mov ah, [CCD_Level]
303 00009BF9 803D[D5410100]CD <1> cmp byte [CD_COMMAND], 0CDh ; 'CD' command or another
304 00009C00 7509 <1> jne short loc_ccd_load_child_dir_err
305 <1> ; It is better to save recent successful part
306 <1> ; of the (requested) path as current directory.
307 <1> ; (Otherwise the path would be reset to back
308 <1> ; on the next 'CD' command.)
309 00009C02 88E1 <1> mov cl, ah
310 00009C04 50 <1> push eax
311 00009C05 E8E3000000 <1> call loc_ccd_save_current_dir
312 00009C0A 58 <1> pop eax
313 <1> loc_ccd_load_child_dir_err:
314 00009C0B 3C03 <1> cmp al, 3 ; AL = 2 => File not found error
315 00009C0D 7202 <1> jb short loc_ccd_path_not_found_retn
316 00009C0F F9 <1> stc
317 00009C10 C3 <1> retn
318 <1>
319 <1> loc_ccd_path_not_found_retn:
320 00009C11 B003 <1> mov al, 3 ; Path not found
321 00009C13 C3 <1> retn
322 <1>
323 <1> loc_ccd_load_FAT_root_dir:
324 00009C14 803D[E5380100]02 <1> cmp byte [Current_FATType], 2
325 00009C1B 776B <1> ja short loc_ccd_load_FAT32_root_dir
326 <1>
327 <1> ;mov esi, [CCD_DriveDT]
328 <1> ;push esi
329 00009C1D E8B61D0000 <1> call load_FAT_root_directory
330 <1> ;pop edi ; Dos Drv Description Table
331 <1>
332 00009C22 89F7 <1> mov edi, esi
333 00009C24 BE[47410100] <1> mov esi, PATH_Array
334 00009C29 7297 <1> jc short loc_ccd_retn_p
335 <1>
336 00009C2B 31C0 <1> xor eax, eax
337 00009C2D EB78 <1> jmp short loc_ccd_set_cdffc
338 <1>
339 <1> loc_ccd_load_root_dir:
340 00009C2F 803D[E5380100]01 <1> cmp byte [Current_FATType], 1
341 00009C36 73DC <1> jnb short loc_ccd_load_FAT_root_dir
342 <1>
343 <1> loc_ccd_load_FS_root_dir:
344 00009C38 E8621E0000 <1> call load_FS_root_directory
345 00009C3D EB5C <1> jmp short pass_ccd_load_FAT_sub_directory
346 <1>
347 <1> loc_ccd_load_FS_sub_directory_next:
348 00009C3F E85C1E0000 <1> call load_FS_sub_directory
349 00009C44 EB1F <1> jmp short pass_ccd_set_dir_cluster_ptr
350 <1>
351 <1> loc_ccd_set_dir_cluster_ptr:
352 <1> ; EDI = Directory Entry
353 00009C46 668B4714 <1> mov ax, [edi+20] ; First Cluster High Word
354 00009C4A C1E010 <1> shl eax, 16
355 00009C4D 668B471A <1> mov ax, [edi+26] ; First Cluster Low Word
356 <1>
357 00009C51 8B35[C7410100] <1> mov esi, [CCD_DriveDT]
358 00009C57 803D[E5380100]01 <1> cmp byte [Current_FATType], 1
359 00009C5E 72DF <1> jb short loc_ccd_load_FS_sub_directory_next
360 <1> ;push esi
361 00009C60 E8FE1D0000 <1> call load_FAT_sub_directory
362 <1> ;pop edi ; Dos Drv Description Table
363 <1>
364 <1> pass_ccd_set_dir_cluster_ptr:
365 <1> ;mov edi, esi
366 00009C65 BE[47410100] <1> mov esi, PATH_Array
367 00009C6A 7264 <1> jc short loc_ccd_retn_c
368 <1>
369 00009C6C A1[15410100] <1> mov eax, [DirBuff_Cluster]
370 <1>
371 00009C71 FE05[CB410100] <1> inc byte [CCD_Level]
372 00009C77 0FB61D[CB410100] <1> movzx ebx, byte [CCD_Level]
373 00009C7E C0E304 <1> shl bl, 4 ; * 16 (<= 128)
374 00009C81 01DE <1> add esi, ebx ; 19/02/2016
375 00009C83 89460C <1> mov [esi+12], eax
376 00009C86 EB1F <1> jmp short loc_ccd_set_cdffc
377 <1>
378 <1> loc_ccd_load_FAT32_root_dir:
379 00009C88 BE[47410100] <1> mov esi, PATH_Array
380 00009C8D 8B460C <1> mov eax, [esi+12]
381 00009C90 8B35[C7410100] <1> mov esi, [CCD_DriveDT]
382 <1>
383 <1> loc_ccd_load_FAT_sub_directory:
384 <1> ;push esi
385 00009C96 E8C81D0000 <1> call load_FAT_sub_directory

```

```

386          <1>      ;pop    edi ; Dos Drv Description Table
387          <1>
388          <1> pass_ccd_load_FAT_sub_directory:
389          <1>      ;mov    edi, esi
390 00009C9B BE[47410100] <1>      mov    esi, PATH_Array
391 00009CA0 722E <1>      jc     short loc_ccd_retn_c
392          <1>
393 00009CA2 A1[15410100] <1>      mov    eax, [DirBuff_Cluster]
394          <1>
395          <1> loc_ccd_set_cdfc:
396 00009CA7 8A0D[CB410100] <1>      mov    cl, [CCD_Level]
397 00009CAD 880D[E4380100] <1>      mov    [Current_Dir_Level], cl
398 00009CB3 A3[E0380100] <1>      mov    [Current_Dir_FCluster], eax
399          <1>
400 00009CB8 8A2D[CC410100] <1>      mov    ch, [Last_Dir_Level]
401 00009CBE 38E9 <1>      cmp    cl, ch
402 00009CC0 0F821CFFFFFF <1>      jnb    loc_ccd_load_child_dir_next
403          <1>
404 00009CC6 803D[D5410100]CD <1>      cmp    byte [CD_COMMAND], 0CDh ; 'CD' command or another
405 00009CCD 741E <1>      je     short loc_ccd_save_current_dir
406          <1>
407          <1>      ; jne -> don't save, restore (the previous cdir) later !
408          <1>      ; (saving the cdir would prevent previous cdir restoration!)
409          <1>
410 00009CCF F8 <1>      cld
411          <1>
412          <1> loc_ccd_retn_c:
413 00009CD0 8B3D[C7410100] <1>      mov    edi, [CCD_DriveDT]
414 00009CD6 C3 <1>      retn
415          <1>
416          <1> loc_ccd_load_sub_directory:
417 00009CD7 8B35[C7410100] <1>      mov    esi, [CCD_DriveDT]
418 00009CDD 803D[E5380100]01 <1>      cmp    byte [Current_FATType], 1
419 00009CE4 73B0 <1>      jnb    short loc_ccd_load_FAT_sub_directory
420 00009CE6 E8B51D0000 <1>      call   load_FS_sub_directory
421 00009CEB EBAB <1>      jmp     short pass_ccd_load_FAT_sub_directory
422          <1>
423          <1> loc_ccd_save_current_dir:
424 00009CED BE[47410100] <1>      mov    esi, PATH_Array ; 19/02/2016
425 00009CF2 8B3D[C7410100] <1>      mov    edi, [CCD_DriveDT]
426 00009CF8 57 <1>      push   edi
427 00009CF9 83C77F <1>      add     edi, LD_CDirLevel
428 00009CFC 880F <1>      mov    [edi], cl
429 00009CFE 47 <1>      inc    edi ; LD_CurrentDirectory
430 00009CFF 56 <1>      push   esi
431          <1>      ;mov    ecx, 32 ; always < 65536 (in this procedure)
432 00009D00 66B92000 <1>      mov    cx, 32
433 00009D04 F3A5 <1>      rep    movsd
434          <1>      ; Current directory has been saved to
435          <1>      ; the DOS drive description table, cdir area !
436 00009D06 5E <1>      pop    esi ; PATH_Array
437 00009D07 5F <1>      pop    edi ; Dos Drv Description Table
438          <1>
439 00009D08 C3 <1>      retn
440          <1>
441          <1> parse_dir_name:
442          <1>      ; 11/02/2016
443          <1>      ; 10/02/2016
444          <1>      ; 07/02/2016 (TRDOS 386 = TRDOS v2.0)
445          <1>      ; 18/09/2011
446          <1>      ; 17/10/2009
447          <1>      ; INPUT ->
448          <1>      ;     ESI = ASCIIZ Directory String Address
449          <1>      ;     AL = Current Directory Level
450          <1>      ;     EDI = Destination Address
451          <1>      ;     (8 levels, each one 12+4 byte)
452          <1>      ; OUTPUT ->
453          <1>      ;     EDI = Dir Entry Formatted Array
454          <1>      ;     with zero cluster pointer at the last level
455          <1>      ;     AH = Last Dir Level
456          <1>      ;     AL = Current Dir Level
457          <1>      ;
458          <1>      ; (esi, ebx, ecx will be changed)
459          <1>
460          <1>      ;mov    [PATH_Array_Ptr], edi
461 00009D09 88C4 <1>      mov    ah, al
462 00009D0B 66A3[6C420100] <1>      mov    [PATH_CDLevel], ax
463          <1> repeat_ppdn_check_slash:
464 00009D11 AC <1>      lodsb
465 00009D12 3C2F <1>      cmp    al, '/'
466 00009D14 74FB <1>      je     short repeat_ppdn_check_slash
467 00009D16 3C21 <1>      cmp    al, 21h
468 00009D18 7219 <1>      jnb    short loc_ppdn_retn
469 00009D1A 57 <1>      push   edi
470          <1> loc_ppdn_get_dir_name:
471 00009D1B B90C000000 <1>      mov    ecx, 12
472 00009D20 BF[6E420100] <1>      mov    edi, Dir_File_Name
473          <1> repeat_ppdn_get_dir_name:
474 00009D25 AA <1>      stosb
475 00009D26 AC <1>      lodsb
476 00009D27 3C2F <1>      cmp    al, '/'
477 00009D29 740A <1>      je     short loc_check_level_dot_conv_dir_name
478 00009D2B 3C20 <1>      cmp    al, 20h
479 00009D2D 7605 <1>      jna    short loc_ppdn_end_of_path_scan
480 00009D2F E2F4 <1>      loop   repeat_ppdn_get_dir_name
481 00009D31 5F <1>      pop    edi
482 00009D32 F9 <1>      stc
483          <1> loc_ppdn_retn:
484 00009D33 C3 <1>      retn
485          <1>
486          <1> loc_ppdn_end_of_path_scan:
487 00009D34 4E <1>      dec    esi

```



```

488                                     <1> loc_check_level_dot_conv_dir_name:
489 00009D35 31C0                       <1>     xor     eax, eax
490 00009D37 AA                         <1>     stosb
491 00009D38 89F3                       <1>     mov     ebx, esi
492 00009D3A BE[6E420100]               <1>     mov     esi, Dir_File_Name
493 00009D3F AC                         <1>     lodsb
494                                     <1> repeat_ppdn_name_check_dot:
495 00009D40 3C2E                       <1>     cmp     al, '.'
496 00009D42 7509                       <1>     jne     short loc_ppdn_convert_sub_dir_name
497                                     <1> repeat_ppdn_name_dot_dot:
498 00009D44 AC                         <1>     lodsb
499 00009D45 3C2E                       <1>     cmp     al, '.'
500 00009D47 743E                       <1>     je      short loc_ppdn_dot_dot
501 00009D49 3C21                       <1>     cmp     al, 21h
502 00009D4B 7226                       <1>     jnb     short pass_ppdn_convert_sub_dir_name
503                                     <1> loc_ppdn_convert_sub_dir_name:
504 00009D4D 8A25[6D420100]               <1>     mov     ah, [PATH_Level]
505 00009D53 80FC07                     <1>     cmp     ah, 7
506 00009D56 731B                       <1>     jnb     short pass_ppdn_convert_sub_dir_name
507 00009D58 FEC4                       <1>     inc     ah
508 00009D5A 8825[6D420100]               <1>     mov     [PATH_Level], ah
509 00009D60 BE[6E420100]               <1>     mov     esi, Dir_File_Name
510                                     <1>     ;mov     edi, [PATH_Array_Ptr]
511 00009D65 B010                       <1>     mov     al, 16
512 00009D67 F6E4                       <1>     mul     ah
513 00009D69 8B3C24                     <1>     mov     edi, [esp]
514                                     <1>     ;push     edi
515 00009D6C 01C7                       <1>     add     edi, eax
516 00009D6E E828030000                 <1>     call    convert_file_name
517                                     <1>     ;pop     edi
518                                     <1> pass_ppdn_convert_sub_dir_name:
519 00009D73 89DE                       <1>     mov     esi, ebx
520                                     <1> repeat_ppdn_check_last_slash:
521 00009D75 AC                         <1>     lodsb
522 00009D76 3C2F                       <1>     cmp     al, '/'
523 00009D78 74FB                       <1>     je      short repeat_ppdn_check_last_slash
524 00009D7A 3C21                       <1>     cmp     al, 21h
525 00009D7C 739D                       <1>     jnb     short loc_ppdn_get_dir_name
526                                     <1> end_of_parse_dir_name:
527 00009D7E 5F                         <1>     pop     edi
528 00009D7F F5                         <1>     cmc
529                                     <1>     ;mov     al, [PATH_CDLevel]
530                                     <1>     ;mov     ah, [PATH_Level]
531 00009D80 66A1[6C420100]               <1>     mov     ax, [PATH_CDLevel]
532 00009D86 C3                         <1>     retn
533                                     <1>
534                                     <1> loc_ppdn_dot_dot:
535 00009D87 AC                         <1>     lodsb
536 00009D88 3C21                       <1>     cmp     al, 21h
537 00009D8A 73F2                       <1>     jnb     short end_of_parse_dir_name
538                                     <1> loc_ppdn_dot_dot_prev_level:
539 00009D8C 66A1[6C420100]               <1>     mov     ax, [PATH_CDLevel]
540 00009D92 80EC01                     <1>     sub     ah, 1
541 00009D95 80D400                     <1>     adc     ah, 0
542 00009D98 38E0                       <1>     cmp     al, ah
543 00009D9A 7602                       <1>     jna     short pass_ppdn_set_al_to_ah
544 00009D9C 88E0                       <1>     mov     al, ah
545                                     <1> pass_ppdn_set_al_to_ah:
546 00009D9E 66A3[6C420100]               <1>     mov     [PATH_CDLevel], ax
547 00009DA4 EBCD                       <1>     jmp     short pass_ppdn_convert_sub_dir_name
548                                     <1>
549                                     <1> locate_current_dir_file:
550                                     <1>     ; 14/02/2016
551                                     <1>     ; 13/02/2016
552                                     <1>     ; 10/02/2016
553                                     <1>     ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
554                                     <1>     ; 14/08/2010
555                                     <1>     ; 19/09/2009
556                                     <1>     ; 2005
557                                     <1>     ; INPUT ->
558                                     <1>     ;     ESI = DOS DirEntry Format FileName Address
559                                     <1>     ;     AL = Attributes Mask
560                                     <1>     ;     (<AL AND EntryAttrib> must be equal to AL)
561                                     <1>     ;     AH = Negative Attributes Mask (If AH>0)
562                                     <1>     ;     (<AH AND EntryAttrib> must be ZERO)
563                                     <1>     ;     CH > 0 Find First Free Dir Entry or Deleted Entry
564                                     <1>     ;     CL = 0 -> Return the First Free Dir Entry
565                                     <1>     ;     CL = E5h -> Return the 1st deleted entry
566                                     <1>     ;     CL = FFh -> Return the 1st deleted or free entry
567                                     <1>     ;     CL > 0 and CL <> E5h and CL <> FFh -> Return the first
568                                     <1>     ;     proper entry (which fits with Attributes Masks)
569                                     <1>     ;     CX = 0 Find Valid File/Directory/VolumeName
570                                     <1>     ;     ? = Any One Char
571                                     <1>     ;     * = Every Chars
572                                     <1>     ; OUTPUT ->
573                                     <1>     ;     EDI = Directory Entry Address (in Directory Buffer)
574                                     <1>     ;     ESI = DOS DirEntry Format FileName Address
575                                     <1>     ;     CF = 0 -> No Error, Proper Entry,
576                                     <1>     ;     DL = Attributes
577                                     <1>     ;     DH = Previous Entry Attr (LongName Check)
578                                     <1>     ;     AL > 0 -> Ambiguous filename wildcard "?" used
579                                     <1>     ;     AH > 0 -> Ambiguous filename wildcard "*" used
580                                     <1>     ;     AX = 0 -> Filename full fits with directory entry
581                                     <1>     ;     CH = The 1st Name Char of Current Dir Entry
582                                     <1>     ;     CF = 1 -> Proper entry not found, Error Code in EAX/AL
583                                     <1>     ;     CL = 0 and CH = 0 -> Free Entry (End Of Dir)
584                                     <1>     ;     CL = 0 and CH = E5h -> Deleted Entry fits with filters
585                                     <1>     ;     CL > 0 -> Entry not found, CH invalid
586                                     <1>     ;     CF = 0 ->
587                                     <1>     ;     EBX = Current Directory Entry Index/Number (BX)
588                                     <1>
589                                     <1>     ;mov     word [DirBuff_EntryCounter], 0 ; Zero Based

```

```

590                                     <1>
591 00009DA6 8935[CF410100]           <1>      mov     [CDLF_FNAddress], esi
592 00009DAC 66A3[CD410100]           <1>      mov     [CDLF_AttributesMask], ax
593 00009DB2 66890D[D3410100]         <1>      mov     [CDLF_DEType], cx
594                                     <1>
595 00009DB9 31DB                       <1>      xor     ebx, ebx
596 00009DBB 881D[E4410100]           <1>      mov     [PreviousAttr], bl ; 0 ; 13/02/2016
597                                     <1>
598 00009DC1 8A3D[E6380100]           <1>      mov     bh, [Current_Drv]
599 00009DC7 381D[10410100]           <1>      cmp     byte [DirBuff_ValidData], bl ; 0
600 00009DCD 761D                       <1>      jna     short loc_lcdf_reload_current_dir2
601 00009DCF 8A1D[0E410100]           <1>      mov     bl, [DirBuff_DRV]
602 00009DD5 80EB41                   <1>      sub     bl, 'A'
603 00009DD8 38DF                       <1>      cmp     bh, bl
604 00009DDA 750E                       <1>      jne     short loc_lcdf_reload_current_dir1
605 00009DDC 8B15[15410100]           <1>      mov     edx, [DirBuff_Cluster]
606 00009DE2 3B15[E0380100]           <1>      cmp     edx, [Current_Dir_FCluster]
607 00009DE8 7412                       <1>      je      short loc_cdir_locatefile_search
608                                     <1>
609                                     <1> loc_lcdf_reload_current_dir1:
610 00009DEA 30DB                       <1>      xor     bl, bl
611                                     <1> loc_lcdf_reload_current_dir2:
612 00009DEC 89DE                       <1>      mov     esi, ebx
613 00009DEE 81C600010900             <1>      add     esi, Logical_DOSDisks
614 00009DF4 E872000000             <1>      call    reload_current_directory
615 00009DF9 735B                       <1>      jnc     short loc_locatefile_search_again
616 00009DFB C3                       <1>      retn
617                                     <1>
618                                     <1> loc_cdir_locatefile_search:
619 00009DFC 31DB                       <1>      xor     ebx, ebx
620 00009DFE E8A5000000             <1>      call    find_directory_entry
621 00009E03 7349                       <1>      jnc     short loc_cdir_locate_file_retn
622                                     <1>
623                                     <1> loc_locatefile_check_stc_reason:
624 00009E05 08ED                       <1>      or      ch, ch
625 00009E07 7444                       <1>      jz      short loc_cdir_locate_file_stc_retn
626                                     <1>
627                                     <1> loc_locatefile_check_next_entryblock:
628 00009E09 8A3D[E6380100]           <1>      mov     bh, [Current_Drv]
629 00009E0F 28DB                       <1>      sub     bl, bl
630 00009E11 0FB7F3                   <1>      movzx    esi, bx
631 00009E14 81C600010900             <1>      add     esi, Logical_DOSDisks
632                                     <1>
633 00009E1A 803D[E4380100]00         <1>      cmp     byte [Current_Dir_Level], 0
634 00009E21 760A                       <1>      jna     short loc_locatefile_check_FAT_type
635                                     <1>
636 00009E23 803D[E5380100]01         <1>      cmp     byte [Current_FATType], 1
637 00009E2A 730A                       <1>      jnb     short loc_locatefile_load_subdir_cluster
638 00009E2C C3                       <1>      retn
639                                     <1>
640                                     <1> loc_locatefile_check_FAT_type:
641 00009E2D 803D[E5380100]03         <1>      cmp     byte [Current_FATType], 3
642 00009E34 7218                       <1>      jb      short loc_cdir_locate_file_retn
643                                     <1>
644                                     <1> loc_locatefile_load_subdir_cluster:
645 00009E36 A1[15410100]             <1>      mov     eax, [DirBuff_Cluster]
646 00009E3B E83D1A0000             <1>      call    get_next_cluster
647 00009E40 730D                       <1>      jnc     short loc_locatefile_next_cluster
648 00009E42 09C0                       <1>      or      eax, eax
649 00009E44 7507                       <1>      jnz     short loc_locatefile_drive_not_ready_read_err
650 00009E46 F9                       <1>      stc
651                                     <1> loc_locatefile_file_notfound:
652 00009E47 B802000000             <1>      mov     eax, 2 ; File/Directory/VolName not found
653 00009E4C C3                       <1>      retn
654                                     <1>
655                                     <1> loc_locatefile_drive_not_ready_read_err:
656                                     <1>      ;mov     eax, 17 ;Drive not ready or read error
657                                     <1> loc_cdir_locate_file_stc_retn:
658 00009E4D F5                       <1>      cmc ;stc
659                                     <1> loc_cdir_locate_file_retn:
660 00009E4E C3                       <1>      retn
661                                     <1>
662                                     <1> loc_locatefile_next_cluster:
663 00009E4F E80F1C0000             <1>      call    load_FAT_sub_directory
664                                     <1>      ;jc      short loc_locatefile_drive_not_ready_read_err
665 00009E54 72F8                       <1>      jc      short loc_cdir_locate_file_retn
666                                     <1>
667                                     <1> loc_locatefile_search_again:
668 00009E56 8B35[CF410100]           <1>      mov     esi, [CDLF_FNAddress]
669 00009E5C 66A1[CD410100]           <1>      mov     ax, [CDLF_AttributesMask]
670 00009E62 668B0D[D3410100]         <1>      mov     cx, [CDLF_DEType]
671 00009E69 EB91                       <1>      jmp     short loc_cdir_locatefile_search
672                                     <1>
673                                     <1> reload_current_directory:
674                                     <1>      ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
675                                     <1>      ; 13/06/2010
676                                     <1>      ; 22/09/2009
677                                     <1>      ;
678                                     <1>      ; INPUT ->
679                                     <1>      ; ESI = Dos drive description table address
680                                     <1>
681                                     <1>      ;mov     al, [esi+LD_FATType]
682 00009E6B A0[E5380100]           <1>      mov     al, [Current_FATType]
683 00009E70 3C02                       <1>      cmp     al, 2
684 00009E72 7729                       <1>      ja      short loc_reload_FAT_sub_directory
685 00009E74 8A25[E4380100]           <1>      mov     ah, [Current_Dir_Level]
686 00009E7A 08C0                       <1>      or      al, al
687 00009E7C 740A                       <1>      jz      short loc_reload_FS_directory
688 00009E7E 08E4                       <1>      or      ah, ah
689 00009E80 751B                       <1>      jnz     short loc_reload_FAT_sub_directory
690                                     <1> loc_reload_FAT_12_16_root_directory:
691 00009E82 E8511B0000             <1>      call    load_FAT_root_directory

```

```

692 00009E87 C3          <1>      retn
693                    <1> loc_reload_FS_directory:
694 00009E88 20E4        <1>      and     ah, ah
695 00009E8A 7506        <1>      jnz     short loc_reload_FS_sub_directory
696                    <1> loc_reload_FS_root_directory:
697 00009E8C E80E1C0000   <1>      call    load_FS_root_directory
698 00009E91 C3          <1>      retn
699                    <1> loc_reload_FS_sub_directory:
700 00009E92 A1[E0380100] <1>      mov     eax, [Current_Dir_FCluster]
701 00009E97 E8041C0000   <1>      call    load_FS_sub_directory
702 00009E9C C3          <1>      retn
703                    <1> loc_reload_FAT_sub_directory:
704 00009E9D A1[E0380100] <1>      mov     eax, [Current_Dir_FCluster]
705 00009EA2 E8BC1B0000   <1>      call    load_FAT_sub_directory
706 00009EA7 C3          <1>      retn
707                    <1>
708                    <1> find_directory_entry:
709                    <1>      ; 14/02/2016
710                    <1>      ; 13/02/2016
711                    <1>      ; 10/02/2016
712                    <1>      ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
713                    <1>      ; 14/08/2010 (DIR.ASM, "proc_find_direntry")
714                    <1>      ; 19/09/2009
715                    <1>      ; 2005
716                    <1>      ; INPUT ->
717                    <1>      ;     ESI = Sub Dir or File Name Address
718                    <1>      ;     AL = Attributes Mask
719                    <1>      ;     (<AL AND EntryAttrib> must be equal to AL)
720                    <1>      ;     AH = Negative Attributes Mask (If AH>0)
721                    <1>      ;     (<AH AND EntryAttrib> must be ZERO)
722                    <1>      ;     CH > 0 Find First Free Dir Entry or Deleted Entry
723                    <1>      ;     CL = 0 -> Return the First Free Dir Entry
724                    <1>      ;     CL = E5h -> Return the 1st deleted entry
725                    <1>      ;     CL = FFh -> Return the 1st deleted or free entry
726                    <1>      ;     CL > 0 and CL <> E5h and CL <> FFh -> Return the first
727                    <1>      ;         proper entry (which fits with Attributes Masks)
728                    <1>      ;     CX = 0 -> Find Valid File/Directory/VolumeName
729                    <1>      ;     ? = Any One Char
730                    <1>      ;     * = Every Chars
731                    <1>      ;     EBX = Current Dir Entry (BX)
732                    <1>      ;
733                    <1>      ; OUTPUT ->
734                    <1>      ;     EDI = Directory Entry Address (in DirectoryBuffer)
735                    <1>      ;     ESI = Sub Dir or File Name Address
736                    <1>      ;     CF = 0 -> No Error, Proper Entry,
737                    <1>      ;     DL = Attributes
738                    <1>      ;     DH = Previous Entry Attr (LongName Check)
739                    <1>      ;     AL > 0 -> Ambiguous filename wildcard "?" used
740                    <1>      ;     AH > 0 -> Ambiguous filename wildcard "*" used
741                    <1>      ;     AX = 0 -> Filename full fits with directory entry
742                    <1>      ;     EBX = CurrentDirEntry (BX)
743                    <1>      ;     CH = The 1st Name Char of Current Dir Entry
744                    <1>      ;     CF = 1 -> Proper entry not found, Error Code in AX/AL
745                    <1>      ;     CL = 0 and CH = 0 -> Free Entry (End Of Dir)
746                    <1>      ;     CL = 0 and CH = E5h -> Deleted Entry fits with filters
747                    <1>      ;     CL > 0 -> Entry not found, CH invalid
748                    <1>      ;
749                    <1>      ; (EAX, EBX, ECX, EDX, EDI, EBP will be changed)
750                    <1>
751 00009EA8 663B1D[13410100] <1>      cmp     bx, [DirBuff_LastEntry]
752 00009EAF 0F8739010000   <1>      ja      loc_ffde_stc_retn_255
753                    <1>
754                    <1>      ;mov     [DirBuff_CurrentEntry], bx
755                    <1>
756 00009EB5 BF00000800   <1>      mov     edi, Directory_Buffer
757 00009EBA 66A3[E0410100] <1>      mov     [FDE_AttrMask], ax
758                    <1>
759 00009EC0 29C0          <1>      sub     eax, eax
760                    <1>
761                    <1>      ;;mov [PreviousAttr], al ; 0 ;; 13/02/2016
762 00009EC2 66A3[E2410100] <1>      mov     [AmbiguousFileName], ax ; 0
763                    <1>
764 00009EC8 6689D8        <1>      mov     ax, bx
765 00009ECB 66C1E005        <1>      shl     ax, 5 ; ; * 32 ; Directory entry size
766 00009ECF 01C7          <1>      add     edi, eax
767                    <1>
768 00009ED1 08ED          <1>      or      ch, ch
769 00009ED3 0F852C010000   <1>      jnz     loc_find_free_deleted_entry_0
770                    <1>
771 00009ED9 08C9          <1>      or      cl, cl
772 00009EDB 0F85D0010000   <1>      jnz     loc_ffde_stc_retn_255
773                    <1>
774                    <1> check_find_dir_entry:
775 00009EE1 66A1[E0410100] <1>      mov     ax, [FDE_AttrMask]
776 00009EE7 8A2F          <1>      mov     ch, [edi]
777 00009EE9 80FD00        <1>      cmp     ch, 0 ; Is it never used entry?
778 00009EEC 0F86FF000000   <1>      jna     loc_find_direntry_stc_retn
779 00009EF2 56            <1>      push    esi
780 00009EF3 8A570B        <1>      mov     dl, [edi+0Bh] ; File attributes
781 00009EF6 80FDE5        <1>      cmp     ch, 0E5h ; Is it a deleted file?
782 00009EF9 746D          <1>      je      short loc_find_dir_next_entry_prevdeleted
783                    <1>
784 00009EFB 80FA0F        <1>      cmp     dl, 0Fh ; longname sub component check
785 00009EFE 7505          <1>      jne     short loc_check_attributes_mask
786 00009F00 E8ED010000   <1>      call    save_longname_sub_component
787                    <1>
788                    <1> loc_check_attributes_mask:
789 00009F05 88C6          <1>      mov     dh, al
790 00009F07 20D6          <1>      and     dh, dl
791 00009F09 38F0          <1>      cmp     al, dh
792 00009F0B 0F85BA000000   <1>      jne     loc_find_dir_next_entry
793 00009F11 20D4          <1>      and     ah, dl

```

```

794 00009F13 0F85B2000000 <1> jnz loc_find_dir_next_entry
795 00009F19 80FA0F <1> cmp dl, 0Fh
796 00009F1C 751A <1> jne short pass_dirententry_attr_check
797 <1>
798 00009F1E 3C0F <1> cmp al, 0Fh ; AL = 0Fh -> find long name
799 00009F20 0F85A5000000 <1> jne loc_find_dir_next_entry
800 <1>
801 00009F26 5E <1> pop esi
802 00009F27 6631C0 <1> xor ax, ax
803 00009F2A 8A35[E4410100] <1> mov dh, [PreviousAttr]
804 00009F30 66891D[11410100] <1> mov [DirBuff_CurrentEntry], bx
805 00009F37 C3 <1> retn
806 <1>
807 <1> pass_dirententry_attr_check:
808 00009F38 89FD <1> mov ebp, edi ; 14/02/2016
809 00009F3A B908000000 <1> mov ecx, 8
810 <1> loc_lodsb_find_dir:
811 00009F3F AC <1> lodsb
812 00009F40 3C2A <1> cmp al, '*'
813 00009F42 7508 <1> jne short pass_fde_ambiguous1_check
814 00009F44 FE05[E3410100] <1> inc byte [AmbiguousFileName+1]
815 00009F4A EB28 <1> jmp short loc_check_dirententry_extension
816 <1>
817 <1> pass_fde_ambiguous1_check:
818 00009F4C 3C3F <1> cmp al, '?'
819 00009F4E 750D <1> jne short pass_fde_ambiguous2_check
820 00009F50 FE05[E2410100] <1> inc byte [AmbiguousFileName]
821 00009F56 803F20 <1> cmp byte [edi], 20h
822 00009F59 764E <1> jna short loc_find_dir_next_entry_ebp
823 00009F5B EB14 <1> jmp short loc_scasb_find_dir_inc_di
824 <1>
825 <1> pass_fde_ambiguous2_check:
826 00009F5D 3C20 <1> cmp al, 20h
827 00009F5F 750C <1> jne short loc_scasb_find_dir
828 00009F61 803F20 <1> cmp byte [edi], 20h
829 00009F64 7543 <1> jne short loc_find_dir_next_entry_ebp
830 00009F66 EB0C <1> jmp short loc_check_dirententry_extension
831 <1>
832 <1> loc_find_dir_next_entry_prevdeleted:
833 00009F68 80CA80 <1> or dl, 80h ; Bit 7 -> deleted entry sign
834 00009F6B EB5E <1> jmp short loc_find_dir_next_entry
835 <1>
836 <1> loc_scasb_find_dir:
837 00009F6D 3A07 <1> cmp al, [edi]
838 00009F6F 7538 <1> jne short loc_find_dir_next_entry_ebp
839 <1> loc_scasb_find_dir_inc_di:
840 00009F71 47 <1> inc edi
841 00009F72 E2CB <1> loop loc_lodsb_find_dir
842 <1>
843 <1> loc_check_dirententry_extension:
844 00009F74 BE08000000 <1> mov esi, 8
845 00009F79 89F7 <1> mov edi, esi ; 8
846 00009F7B 033424 <1> add esi, [esp] ; Sub Dir or File Name Address
847 00009F7E 01EF <1> add edi, ebp
848 00009F80 B103 <1> mov cl, 3
849 <1> loc_lodsb_find_dir_ext:
850 00009F82 AC <1> lodsb
851 00009F83 3C2A <1> cmp al, '*'
852 00009F85 7508 <1> jne short pass_fde_ambiguous3_check
853 00009F87 FE05[E3410100] <1> inc byte [AmbiguousFileName+1]
854 00009F8D EB1E <1> jmp short loc_find_dir_proper_dirententry
855 <1>
856 <1> pass_fde_ambiguous3_check:
857 00009F8F 3C3F <1> cmp al, '?'
858 00009F91 750D <1> jne short pass_fde_ambiguous4_check
859 00009F93 FE05[E2410100] <1> inc byte [AmbiguousFileName]
860 00009F99 803F20 <1> cmp byte [edi], 20h
861 00009F9C 760B <1> jna short loc_find_dir_next_entry_ebp
862 00009F9E EB49 <1> jmp short loc_scasb_find_dir_ext_inc_di
863 <1>
864 <1> pass_fde_ambiguous4_check:
865 00009FA0 3C20 <1> cmp al, 20h
866 00009FA2 7541 <1> jne short loc_scasb_find_dir_ext
867 00009FA4 803F20 <1> cmp byte [edi], 20h
868 00009FA7 7404 <1> je short loc_find_dir_proper_dirententry
869 <1>
870 <1> loc_find_dir_next_entry_ebp:
871 00009FA9 89EF <1> mov edi, ebp ; 14/02/2016
872 00009FAB EB1E <1> jmp short loc_find_dir_next_entry
873 <1>
874 <1> loc_find_dir_proper_dirententry:
875 00009FAD 30C9 <1> xor cl, cl
876 <1> loc_find_dir_proper_dirententry_1:
877 00009FAF 5E <1> pop esi
878 00009FB0 89EF <1> mov edi, ebp
879 00009FB2 8A2F <1> mov ch, [edi]
880 00009FB4 8A570B <1> mov dl, [edi+0Bh] ; Dir entry attributes
881 00009FB7 66A1[E2410100] <1> mov ax, [AmbiguousFileName]
882 <1> loc_find_dir_proper_dirententry_2:
883 00009FBD 8A35[E4410100] <1> mov dh, [PreviousAttr]
884 00009FC3 66891D[11410100] <1> mov [DirBuff_CurrentEntry], bx
885 00009FCA C3 <1> retn
886 <1>
887 <1> loc_find_dir_next_entry:
888 00009FCB 8815[E4410100] <1> mov byte [PreviousAttr], dl ; LongName check
889 <1> loc_find_dir_next_entry_1:
890 00009FD1 5E <1> pop esi
891 00009FD2 83C720 <1> add edi, 32
892 <1> ;inc word [DirBuff_EntryCounter]
893 00009FD5 6643 <1> inc bx
894 00009FD7 663B1D[13410100] <1> cmp bx, [DirBuff_LastEntry]
895 00009FDE 770E <1> ja short loc_ffde_stc_retn_255

```



```

896 00009FE0 E9FCFEFFFF      <1>          jmp      check_find_dir_entry
897                          <1>
898                          <1> loc_scasb_find_dir_ext:
899 00009FE5 3A07            <1>          cmp      al, [edi]
900 00009FE7 75C0            <1>          jne      short loc_find_dir_next_entry_ebp
901                          <1> loc_scasb_find_dir_ext_inc_di:
902 00009FE9 47              <1>          inc      edi
903 00009FEA E296            <1>          loop     loc_lodsb_find_dir_ext
904 00009FEC EBC1            <1>          jmp      short loc_find_dir_proper_direntry_1
905                          <1>
906                          <1> loc_ffde_stc_retn_255:
907                          <1>          ;mov     cx, 0FFFFh
908 00009FEE 31C9            <1>          xor      ecx, ecx
909 00009FF0 49              <1>          dec      ecx ; 0FFFFFFFFh
910                          <1>          ;xor     eax, eax
911                          <1> loc_find_direntry_stc_retn:
912                          <1> loc_check_ffde_retn_1:
913                          <1>          ;mov     ax, 2
914 00009FF1 B802000000        <1>          mov     eax, 2 ; File Not Found
915 00009FF6 8A35[E4410100] <1>          mov     dh, [PreviousAttr]
916 00009FFC 66891D[11410100] <1>          mov     [DirBuff_CurrentEntry], bx
917 0000A003 F9              <1>          stc
918 0000A004 C3              <1>          retn
919                          <1>
920                          <1> loc_find_free_deleted_entry_0:
921 0000A005 66A1[E0410100] <1>          mov     ax, [FDE_AttrMask]
922 0000A00B 8A2F            <1>          mov     ch, [edi]
923 0000A00D 8A570B          <1>          mov     dl, [edi+0Bh] ; File attributes
924 0000A010 08C9            <1>          or      cl, cl
925 0000A012 7407            <1>          jz       short loc_check_ffde_0_repeat
926                          <1>          ;cmp     cl, 0E5h
927                          <1>          ;je      short pass_loc_check_ffde_0_err
928 0000A014 80F9FF          <1>          cmp     cl, 0FFh
929 0000A017 7432            <1>          je       short loc_find_free_deleted_entry_1
930 0000A019 EB4D            <1>          jmp      short pass_loc_check_ffde_0_err
931                          <1>
932                          <1> loc_check_ffde_0_repeat:
933 0000A01B 08ED            <1>          or      ch, ch
934 0000A01D 7511            <1>          jnz      short loc_check_ffde_0_next
935                          <1>
936                          <1> loc_check_ffde_retn_2:
937 0000A01F 6629C0          <1>          sub     ax, ax
938 0000A022 8A35[E4410100] <1>          mov     dh, [PreviousAttr]
939 0000A028 66891D[11410100] <1>          mov     [DirBuff_CurrentEntry], bx
940 0000A02F C3              <1>          retn
941                          <1>
942                          <1> loc_check_ffde_0_next:
943 0000A030 6643            <1>          inc     bx
944 0000A032 83C720          <1>          add     edi, 32
945                          <1>          ;inc     word [DirBuff_EntryCounter]
946                          <1>
947 0000A035 663B1D[13410100] <1>          cmp     bx, [DirBuff_LastEntry]
948 0000A03C 77B0            <1>          ja       short loc_ffde_stc_retn_255
949 0000A03E 8815[E4410100] <1>          mov     [PreviousAttr], dl
950 0000A044 8A2F            <1>          mov     ch, [edi]
951 0000A046 8A570B          <1>          mov     dl, [edi+0Bh] ; file attributes
952 0000A049 EBD0            <1>          jmp      short loc_check_ffde_0_repeat
953                          <1>
954                          <1> loc_find_free_deleted_entry_1:
955 0000A04B 28D2            <1>          sub     dl, dl
956                          <1> loc_find_free_deleted_entry_2:
957 0000A04D 20ED            <1>          and     ch, ch
958 0000A04F 74CE            <1>          jz       short loc_check_ffde_retn_2
959 0000A051 80FDE5          <1>          cmp     ch, 0E5h
960 0000A054 74C9            <1>          je       short loc_check_ffde_retn_2
961 0000A056 6643            <1>          inc     bx
962 0000A058 83C720          <1>          add     edi, 32
963 0000A05B 663B1D[13410100] <1>          cmp     bx, [DirBuff_LastEntry]
964 0000A062 778A            <1>          ja       short loc_ffde_stc_retn_255
965 0000A064 8A2F            <1>          mov     ch, [edi]
966 0000A066 EBE5            <1>          jmp      short loc_find_free_deleted_entry_2
967                          <1>
968                          <1> pass_loc_check_ffde_0_err:
969 0000A068 38CD            <1>          cmp     ch, cl
970 0000A06A 741F            <1>          je       short loc_check_ffde_attr
971                          <1>
972 0000A06C 6643            <1>          inc     bx
973 0000A06E 83C720          <1>          add     edi, 32
974 0000A071 663B1D[13410100] <1>          cmp     bx, [DirBuff_LastEntry]
975 0000A078 0F8770FFFFFF      <1>          ja       loc_ffde_stc_retn_255
976 0000A07E 8815[E4410100] <1>          mov     [PreviousAttr], dl
977 0000A084 8A2F            <1>          mov     ch, [edi]
978 0000A086 8A570B          <1>          mov     dl, [edi+0Bh]
979 0000A089 EBDD            <1>          jmp      short pass_loc_check_ffde_0_err
980                          <1>
981                          <1> loc_check_ffde_attr:
982 0000A08B 88C6            <1>          mov     dh, al
983 0000A08D 20D6            <1>          and     dh, dl
984 0000A08F 38F0            <1>          cmp     al, dh
985 0000A091 759D            <1>          jne      short loc_check_ffde_0_next
986 0000A093 20D4            <1>          and     ah, dl
987 0000A095 7599            <1>          jnz      short loc_check_ffde_0_next
988 0000A097 30C9            <1>          xor     cl, cl
989 0000A099 EB84            <1>          jmp      loc_check_ffde_retn_2
990                          <1>
991                          <1> convert_file_name:
992                          <1>          ; 06/03/2016
993                          <1>          ; 11/02/2016
994                          <1>          ; 07/02/2016 (TRDOS 386 = TRDOS v2.0)
995                          <1>          ; 06/10/2009
996                          <1>          ; 2005
997                          <1>          ;

```

```

998      <1>      ; INPUT ->
999      <1>      ;      ESI = Dot File Name Location
1000     <1>      ;      EDI = Dir Entry Format File Name Location
1001     <1>      ; OUTPUT ->
1002     <1>      ;      EDI = Dir Entry Format File Name Location
1003     <1>      ;      ESI = Dot File Name Location (capitalized)
1004     <1>      ;
1005     <1>      ; (ECX, AL will be changed)
1006     <1>
1007     0000A09B 56      <1>      push    esi
1008     0000A09C 57      <1>      push    edi
1009     <1>
1010     0000A09D B90B000000 <1>      mov     ecx, 11
1011     0000A0A2 B020      <1>      mov     al, 20h
1012     0000A0A4 F3AA      <1>      rep     stosb
1013     <1>
1014     0000A0A6 8B3C24      <1>      mov     edi, [esp]
1015     <1>
1016     0000A0A9 B10C      <1>      mov     cl, 12 ; file name length (max.)
1017     <1>      ; 06/03/2016
1018     <1>      ; Directory entry name limit (11 bytes) check for
1019     <1>      ; 'rename_directory_entry' procedure.
1020     <1>      ; (EDI points to Directory Entry)
1021     <1>      ; (If the file name would not contain a dot
1022     <1>      ; and file name length would be 12, this would cause to
1023     <1>      ; overwrite the attributes byte of the directory entry.)
1024     <1>      ;
1025     0000A0AB B50B      <1>      mov     ch, 11 ; directory entry's name length
1026     <1> loc_check_first_dot:
1027     0000A0AD 8A06      <1>      mov     al, [esi]
1028     0000A0AF 3C2E      <1>      cmp     al, 2Eh
1029     0000A0B1 750C      <1>      jne     short pass_check_first_dot
1030     0000A0B3 8807      <1>      mov     [edi], al
1031     0000A0B5 47        <1>      inc     edi
1032     0000A0B6 46        <1>      inc     esi
1033     0000A0B7 FEC9      <1>      dec     cl
1034     0000A0B9 75F2      <1>      jnz     short loc_check_first_dot
1035     <1>      ;;(ecx <= 12)
1036     <1>      ;;loop loc_check_first_dot
1037     0000A0BB EB30      <1>      jmp     short stop_convert_file
1038     <1>
1039     <1> loc_get_fchar:
1040     0000A0BD 8A06      <1>      mov     al, [esi]
1041     <1> pass_check_first_dot:
1042     0000A0BF 3C61      <1>      cmp     al, 61h ; 'a'
1043     0000A0C1 7208      <1>      jnb     short pass_name_capitalize
1044     0000A0C3 3C7A      <1>      cmp     al, 7Ah ; 'z'
1045     0000A0C5 7704      <1>      ja      short pass_name_capitalize
1046     0000A0C7 24DF      <1>      and     al, 0DFh
1047     0000A0C9 8806      <1>      mov     [esi], al
1048     <1> pass_name_capitalize:
1049     0000A0CB 3C21      <1>      cmp     al, 21h
1050     0000A0CD 721E      <1>      jnb     short stop_convert_file
1051     0000A0CF 3C2E      <1>      cmp     al, 2Eh ; '.'
1052     0000A0D1 750C      <1>      jne     short pass_dot_space
1053     <1> add_dot_space:
1054     0000A0D3 80F904      <1>      cmp     cl, 4
1055     0000A0D6 760E      <1>      jna     short inc_and_loop
1056     0000A0D8 47        <1>      inc     edi
1057     0000A0D9 FECD      <1>      dec     ch ; 06/03/2016
1058     0000A0DB FEC9      <1>      dec     cl
1059     0000A0DD 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     0000A0DF 8807      <1>      mov     [edi], al
1071     <1> loc_after_double_dot:
1072     <1>      ; 06/03/2016
1073     0000A0E1 FECD      <1>      dec     ch ; count down for 11 bytes dir entry limit
1074     0000A0E3 740A      <1>      jz      short stop_convert_file_x
1075     0000A0E5 47        <1>      inc     edi
1076     <1> inc_and_loop:
1077     0000A0E6 FEC9      <1>      dec     cl ; count down for 12 bytes filename limit
1078     0000A0E8 7403      <1>      jz      short stop_convert_file
1079     0000A0EA 46        <1>      inc     esi
1080     <1>      ;;(ecx <= 12)
1081     <1>      ;;loop loc_get_fchar
1082     0000A0EB EBD0      <1>      jmp     short loc_get_fchar
1083     <1>
1084     <1> stop_convert_file:
1085     <1>      ; 06/03/2016
1086     0000A0ED 30ED      <1>      xor     ch, ch
1087     <1>      ; ECX < 256 ; 'find_first_file' -> xor cl, cl
1088     <1> stop_convert_file_x:
1089     0000A0EF 5F        <1>      pop     edi
1090     0000A0F0 5E        <1>      pop     esi
1091     0000A0F1 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      0000A0F2 57      <1>      push    edi
1111      0000A0F3 56      <1>      push    esi
1112      <1>      ;push    ebx
1113      <1>      ;push    ecx
1114      <1>      ;push    edx
1115      0000A0F4 50      <1>      push    eax
1116      <1>
1117      0000A0F5 29C9    <1>      sub     ecx, ecx
1118      <1>      ;sub     eax, eax
1119      0000A0F7 B11A    <1>      mov     cl, 26
1120      <1>
1121      0000A0F9 0FB607  <1>      movzx   eax, byte [edi] ; LDIR_Order
1122      0000A0FC 3C41    <1>      cmp     al, 41h ; 40h (last long entry sign) + 1
1123      0000A0FE 722B    <1>      jnb     short pass_pslnsc_last_long_entry
1124      <1>
1125      0000A100 88C4    <1>      mov     ah, al
1126      0000A102 80EC40  <1>      sub     ah, 40h
1127      0000A105 8825[E6410100] <1>      mov     [LFN_EntryLength], ah
1128      <1>
1129      0000A10B 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      0000A10D 7753    <1>      ja      short loc_pslnsc_retn
1133      <1>
1134      0000A10F 2407    <1>      and     al, 07h ; 0Fh
1135      0000A111 A2[E5410100] <1>      mov     [LongNameFound], al
1136      <1>
1137      0000A116 FEC8    <1>      dec     al
1138      <1>      ;mov     cl, 26
1139      0000A118 F6E1    <1>      mul     cl
1140      <1>
1141      0000A11A 89C6    <1>      mov     esi, eax
1142      0000A11C 01CE    <1>      add     esi, ecx
1143      <1>      ; to make is an ASCII string
1144      <1>      ; with ax+26 bytes length
1145      0000A11E 81C6[E8410100] <1>      add     esi, LongFileName
1146      0000A124 66C7060000 <1>      mov     word [esi], 0
1147      0000A129 EB16    <1>      jmp     short loc_pslsc_move_ldir_name2
1148      <1>
1149      <1> pass_pslnsc_last_long_entry:
1150      0000A12B 3C04    <1>      cmp     al, 04h
1151      0000A12D 7733    <1>      ja      short loc_pslnsc_retn
1152      0000A12F FE0D[E5410100] <1>      dec     byte [LongNameFound]
1153      0000A135 3A05[E5410100] <1>      cmp     al, [LongNameFound]
1154      0000A13B 7525    <1>      jne     short loc_pslnsc_retn
1155      <1>
1156      <1> loc_pslsc_move_ldir_name1:
1157      0000A13D FEC8    <1>      dec     al
1158      <1>      ;mov     cl, 26
1159      0000A13F F6E1    <1>      mul     cl
1160      <1>
1161      <1> loc_pslsc_move_ldir_name2:
1162      0000A141 8A4F0D  <1>      mov     cl, [edi+0Dh] ; long name checksum
1163      0000A144 880D[E7410100] <1>      mov     [LFN_CheckSum], cl
1164      0000A14A 89FE    <1>      mov     esi, edi ; LDIR_Order
1165      0000A14C BF[E8410100] <1>      mov     edi, LongFileName
1166      0000A151 01C7    <1>      add     edi, eax
1167      0000A153 46      <1>      inc     esi
1168      0000A154 B105    <1>      mov     cl, 5 ; chars 1 to 5
1169      0000A156 F366A5  <1>      rep     movsw
1170      0000A159 83C603  <1>      add     esi, 3
1171      0000A15C A5      <1>      movsd   ; char 6 & 7
1172      0000A15D A5      <1>      movsd   ; char 8 & 9
1173      0000A15E A5      <1>      movsd   ; char 10 & 11
1174      0000A15F 46      <1>      inc     esi
1175      0000A160 46      <1>      inc     esi
1176      0000A161 A5      <1>      movsd   ; char 12 & 13
1177      <1>
1178      <1> loc_pslnsc_retn:
1179      0000A162 58      <1>      pop     eax
1180      <1>      ;pop     edx
1181      <1>      ;pop     ecx
1182      <1>      ;pop     ebx
1183      0000A163 5E      <1>      pop     esi
1184      0000A164 5F      <1>      pop     edi
1185      <1>
1186      0000A165 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 ASCIIIZ 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 0000A166 57      <1>      push     edi
1207 0000A167 B914000000 <1>      mov      ecx, 20 ; 80 bytes
1208 0000A16C 31C0     <1>      xor      eax, eax
1209 0000A16E F3AB     <1>      rep      stosd
1210 0000A170 5F      <1>      pop      edi
1211      <1>
1212 0000A171 668B06   <1>      mov      ax, [esi]
1213 0000A174 80FC3A   <1>      cmp      ah, ':'
1214 0000A177 741C     <1>      je       short loc_ppn_change_drive
1215 0000A179 A0[E6380100] <1>      mov      al, [Current_Drv]
1216 0000A17E EB33     <1>      jmp      short pass_ppn_change_drive
1217      <1>
1218      <1> pass_ppn_cdir:
1219 0000A180 8B35[0A430100] <1>      mov      esi, [First_Path_Pos]
1220 0000A186 AC      <1>      lodsb
1221      <1> loc_ppn_get_filename:
1222 0000A187 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 0000A18A B10C     <1>      mov      cl, 12
1226      <1> loc_ppn_get_fnchar_next:
1227 0000A18C AA      <1>      stosb
1228 0000A18D AC      <1>      lodsb
1229 0000A18E 3C21     <1>      cmp      al, 21h
1230 0000A190 7274     <1>      jb       short loc_ppn_clc_return
1231 0000A192 E2F8     <1>      loop     loc_ppn_get_fnchar_next
1232      <1> loc_ppn_return:
1233 0000A194 C3      <1>      retn
1234      <1>
1235      <1> loc_ppn_change_drive:
1236 0000A195 24DF     <1>      and      al, 0DFh
1237 0000A197 2C41     <1>      sub      al, 'A'; A:
1238 0000A199 726F     <1>      jc       short loc_ppn_invalid_drive
1239 0000A19B 3805[A7FA0000] <1>      cmp      [Last_DOS_DiskNo], al
1240 0000A1A1 7267     <1>      jb       short loc_ppn_invalid_drive
1241      <1>
1242 0000A1A3 46      <1>      inc      esi
1243 0000A1A4 46      <1>      inc      esi
1244 0000A1A5 8A26     <1>      mov      ah, [esi]
1245 0000A1A7 80FC21   <1>      cmp      ah, 21h
1246 0000A1AA 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 0000A1AC 8807     <1>      mov      [edi], al ; Drv
1251 0000A1AE 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 0000A1B2 C3      <1>      retn
1257      <1>
1258      <1> pass_ppn_change_drive:
1259 0000A1B3 8935[0A430100] <1>      mov      [First_Path_Pos], esi
1260 0000A1B9 C705[0E430100]0000- <1>      mov      dword [Last_Slash_Pos], 0
1261 0000A1C1 0000     <1>
1262 0000A1C3 AA      <1>      stosb
1263 0000A1C4 8A06     <1>      mov      al, [esi]
1264 0000A1C6 3C2F     <1>      cmp      al, '/'
1265 0000A1C8 7506     <1>      jne      short loc_scan_next_slash_pos
1266 0000A1CA 8935[0E430100] <1>      mov      [Last_Slash_Pos], esi
1267      <1> loc_scan_next_slash_pos:
1268 0000A1D0 46      <1>      inc      esi
1269 0000A1D1 8A06     <1>      mov      al, [esi]
1270 0000A1D3 3C20     <1>      cmp      al, 20h
1271 0000A1D5 77EF     <1>      ja       short loc_scan_ppn_dslash
1272 0000A1D7 833D[0E430100]00 <1>      cmp      dword [Last_Slash_Pos], 0
1273 0000A1DE 76A0     <1>      jna      short pass_ppn_cdir
1274      <1>
1275 0000A1E0 8B0D[0E430100] <1>      mov      ecx, [Last_Slash_Pos]
1276 0000A1E6 8B35[0A430100] <1>      mov      esi, [First_Path_Pos]
1277 0000A1EC 29F1     <1>      sub      ecx, esi
1278 0000A1EE 41      <1>      inc      ecx
1279      <1>      ;cmp     ecx, 64
1280 0000A1EF 80F940   <1>      cmp      cl, 64
1281 0000A1F2 7715     <1>      ja       short loc_ppn_invalid_drive_stc
1282      <1>
1283 0000A1F4 89F8     <1>      mov      eax, edi ; Dest Dir String Location (65 byte)
1284 0000A1F6 F3A4     <1>      rep      movsb
1285      <1>      ;mov     [edi], cl ; 0, End of Dir String
1286 0000A1F8 8B35[0E430100] <1>      mov      esi, [Last_Slash_Pos]
1287 0000A1FE 46      <1>      inc      esi
1288 0000A1FF 89C7     <1>      mov      edi, eax
1289 0000A201 AC      <1>      lodsb
1290 0000A202 3C21     <1>      cmp      al, 21h
1291 0000A204 7381     <1>      jnb      short loc_ppn_get_filename
1292      <1> loc_ppn_clc_return:
1293      <1>      ;clc
1294 0000A206 31C0     <1>      xor      eax, eax
1295 0000A208 C3      <1>      retn
1296      <1>
1297      <1> loc_ppn_invalid_drive_stc:
1298 0000A209 F5      <1>      cmc      ; stc
1299      <1> loc_ppn_invalid_drive:
1300      <1>      ; cf = 1
1301      <1>      ; The Drive Letter/Char < "A" or > "Z"
1302 0000A20A 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      0000A20E 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      0000A20F 66B80008      <1>      mov  ax, 0800h
1340      <1>      ; it must not be volume name or longname
1341      0000A213 E837DDFFFF      <1>      call find_first_file
1342      0000A218 7216      <1>      jc   short loc_fln_retn
1343      <1>
1344      <1> loc_fln_check_FAT_Type:
1345      0000A21A 803D[E5380100]01      <1>      cmp  byte [Current_FATType], 1
1346      0000A221 7306      <1>      jnb  short loc_fln_check_longname_yes_sign
1347      <1>
1348      0000A223 E839000000      <1>      call  get_fs_longname
1349      0000A228 C3      <1>      retn
1350      <1>
1351      <1> loc_fln_check_longname_yes_sign:
1352      0000A229 08FF      <1>      or   bh, bh
1353      0000A22B 7504      <1>      jnz  short loc_fln_check_longnamefound_number
1354      <1> loc_fln_longname_not_found_retn:
1355      0000A22D 31C0      <1>      xor  eax, eax
1356      <1>      ; cf = 1 & al = 0 -> longname not found
1357      0000A22F F9      <1>      stc
1358      <1> loc_fln_retn:
1359      0000A230 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      0000A231 803D[E5410100]01      <1>      cmp  byte [LongNameFound], 1
1370      0000A238 75F3      <1>      jne  short loc_fln_longname_not_found_retn
1371      <1>
1372      <1> loc_fln_calculate_checksum:
1373      0000A23A 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      0000A23F 3805[E7410100]      <1>      cmp  [LFN_CheckSum], al
1384      0000A245 75E6      <1>      jne  short loc_fln_longname_not_found_retn
1385      <1>
1386      0000A247 BE[E8410100]      <1>      mov  esi, LongFileName
1387      0000A24C A0[E5380100]      <1>      mov  al, [Current_FATType]
1388      0000A251 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      0000A252 30C0      <1>      xor     al, al
1413      0000A254 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      0000A259 D0C8      <1>      ror     al, 1 ; 17/10/2009
1422      0000A25B 0206      <1>      add     al, [esi]
1423      0000A25D 46        <1>      inc     esi
1424      <1>      ;add     al, ah
1425      0000A25E E2F9      <1>      loop    loc_next_sum
1426      0000A260 C3        <1>      retn
1427      <1>
1428      <1>      get_fs_longname:
1429      <1>      ; temporary (13/02/2016)
1430      0000A261 31C0      <1>      xor     eax, eax
1431      0000A263 F9        <1>      stc
1432      0000A264 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 = ASCIIIZ 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      0000A265 80E107      <1>      and     cl, 07h
1455      0000A268 880D[64430100] <1>      mov     byte [mkdir_attr], cl
1456      <1>
1457      0000A26E 56        <1>      push    esi
1458      0000A26F 31DB      <1>      xor     ebx, ebx
1459      0000A271 8A3D[E6380100] <1>      mov     bh, [Current_Drv]
1460      0000A277 BE00010900 <1>      mov     esi, Logical_DOSDisks
1461      0000A27C 01DE      <1>      add     esi, ebx
1462      0000A27E 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      0000A27F 807E0101 <1>      cmp     byte [esi+LD_DiskType], 1 ; 0 = Invalid
1467      0000A283 730B      <1>      jnb     short loc_mkdir_check_file_sytem
1468      <1>      ; 16/10/2016 (13h -> 30)
1469      0000A285 B81E000000 <1>      mov     eax, 30 ; 'Disk write-protected' error
1470      0000A28A 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      0000A28F 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      0000A290 807E0301 <1>      cmp     byte [esi+LD_FATType], 1
1482      0000A294 730B      <1>      jnb     short loc_mkdir_check_free_sectors
1483      <1>
1484      <1>      loc_make_fs_directory:
1485      0000A296 A1[E0380100] <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 ASCIIIZ name)
1489      0000A29B E8D8150000 <1>      call    make_fs_directory
1490      0000A2A0 C3        <1>      retn
1491      <1>
1492      <1>      loc_mkdir_check_free_sectors:
1493      0000A2A1 0FB64613 <1>      movzx    eax, byte [esi+LD_BPB+SecPerClust]
1494      0000A2A5 8B4E74      <1>      mov     ecx, [esi+LD_FreeSectors]
1495      0000A2A8 39C1      <1>      cmp     ecx, eax
1496      0000A2AA 7255      <1>      jb      short loc_mkdir_insufficient_disk_space
1497      <1>
1498      <1>      loc_make_fat_directory:
1499      0000A2AC 891D[54430100] <1>      mov     [mkdir_DirName_Offset], ebx
1500      0000A2B2 890D[60430100] <1>      mov     [mkdir_FreeSectors], ecx
1501      <1>
1502      <1>      ;mov     al, [esi+LD_BPB+SecPerClust]
1503      0000A2B8 A2[66430100] <1>      mov     byte [mkdir_SecPerClust], al
1504      <1>
1505      <1>      loc_mkdir_gffc_1:
1506      0000A2BD E812180000 <1>      call    get_first_free_cluster

```

```

1507 0000A2C2 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 0000A2C4 A3[58430100] <1>      mov     [mkdir_FFCluster], eax
1515                  <1>
1516                  <1> loc_mkdir_locate_ffc:
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 0000A2C9 31C0      <1>      xor     eax, eax
1524 0000A2CB 89C1      <1>      mov     ecx, eax
1525 0000A2CD 6649      <1>      dec     cx ; FFFFh
1526                  <1>      ; CX = FFFFh -> find first deleted or free entry
1527                  <1>      ; ESI would be ASCII filename address if the call
1528                  <1>      ; would not be for first free or deleted dir entry
1529 0000A2CF E8D2FAFFFF <1>      call    locate_current_dir_file
1530 0000A2D4 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 0000A2D6 83F802    <1>      cmp     eax, 2 ; cmp al, 2 ; File/Dir not found !
1534 0000A2D9 752B      <1>      jne     short loc_mkdir_stc_return
1535                  <1>
1536                  <1> loc_mkdir_add_new_cluster:
1537 0000A2DB 3805[E5380100] <1>      cmp     byte [Current_FATType], al ; 2
1538                  <1>      ;cmp     byte ptr [esi+LD_FATType], 2
1539 0000A2E1 770C      <1>      ja      short loc_mkdir_add_new_cluster_check_fsc
1540 0000A2E3 803D[E4380100]01 <1>      cmp     byte [Current_Dir_Level], 1
1541                  <1>      ;cmp     byte [esi+LD_CDirLevel], 1
1542 0000A2EA 7303      <1>      jnb     short loc_mkdir_add_new_cluster_check_fsc
1543                  <1>
1544 0000A2EC B00C      <1>      mov     al, 12 ; No more files
1545                  <1> loc_mkdir_gffc_retn:
1546 0000A2EE C3          <1>      retn
1547                  <1>
1548                  <1> loc_mkdir_add_new_cluster_check_fsc:
1549 0000A2EF 8B0D[60430100] <1>      mov     ecx, [mkdir_FreeSectors]
1550                  <1>      ;movzx   eax, byte [mkdir_SecPerClust]
1551 0000A2F5 A0[66430100] <1>      mov     al, [mkdir_SecPerClust]
1552 0000A2FA 66D1E0    <1>      shl     ax, 1 ; AX = 2 * AX
1553 0000A2FD 39C1      <1>      cmp     ecx, eax
1554 0000A2FF 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 0000A301 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 0000A305 C3          <1>      retn
1565                  <1>
1566                  <1> loc_mkdir_stc_return:
1567 0000A306 F9          <1>      stc
1568 0000A307 C3          <1>      retn
1569                  <1>
1570                  <1> loc_mkdir_gffc_2:
1571 0000A308 E8C7170000    <1>      call    get_first_free_cluster
1572 0000A30D 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 0000A30F A3[58430100] <1>      mov     [mkdir_FFCluster], eax
1580                  <1>
1581 0000A314 A1[5C430100] <1>      mov     eax, [mkdir_LastDirCluster]
1582                  <1>
1583 0000A319 E845170000    <1>      call    load_FAT_sub_directory
1584 0000A31E 72CE      <1>      jc      short loc_mkdir_gffc_retn
1585                  <1>
1586 0000A320 31FF      <1>      xor     edi, edi
1587                  <1> loc_mkdir_set_ff_dir_entry_1:
1588                  <1>      ; 27/02/2016
1589 0000A322 56          <1>      push    esi ; Logical DOS Drv Desc. Tbl. address
1590                  <1>      ; EDI = Directory Entry Address
1591 0000A323 8B35[54430100] <1>      mov     esi, [mkdir_DirName_Offset]
1592 0000A329 A1[58430100] <1>      mov     eax, [mkdir_FFCluster]
1593                  <1>
1594 0000A32E 66B91000    <1>      mov     cx, 10h ; CL = Directory attribute
1595                  <1>      ; CH = 0 -> File size is 0
1596 0000A332 0A0D[64430100] <1>      or      cl, [mkdir_attr] ; S, H, R
1597 0000A338 E8B0010000    <1>      call    make_directory_entry
1598                  <1>
1599 0000A33D 5E          <1>      pop     esi
1600                  <1>
1601 0000A33E C605[10410100]02 <1>      mov     byte [DirBuff_ValidData], 2
1602 0000A345 E880020000    <1>      call    save_directory_buffer
1603 0000A34A 0F83DA000000 <1>      jnc     loc_mkdir_set_ff_dir_entry_2
1604                  <1>
1605                  <1> loc_mkdir_return:
1606 0000A350 C3          <1>      retn
1607                  <1>
1608                  <1> loc_mkdir_add_new_subdir_cluster:

```

```

1609 0000A351 8B15[15410100] <1> mov     edx, [DirBuff_Cluster]
1610 0000A357 8915[5C430100] <1> mov     [mkdir_LastDirCluster], edx
1611 <1>
1612 0000A35D A1[58430100] <1> mov     eax, [mkdir_FFCluster]
1613 0000A362 E8FC160000 <1> call    load_FAT_sub_directory
1614 0000A367 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 0000A369 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 0000A36B 668B4611 <1> mov     ax, [esi+LD_BPB+BytesPerSec] ; 512
1625 0000A36F 66C1E802 <1> shr     ax, 2 ; 'byte count / 4' for 'stosd'
1626 0000A373 66F7E1 <1> mul     cx ; max = 128*(512/4) -> 16384 (stosd)
1627 0000A376 6689C1 <1> mov     cx, ax
1628 0000A379 6629C0 <1> sub     ax, ax ; 0
1629 0000A37C F3AB <1> rep     stosd ; clear directory buffer
1630 <1>
1631 0000A37E C605[10410100]02 <1> mov     byte [DirBuff_ValidData], 2
1632 0000A385 E840020000 <1> call    save_directory_buffer
1633 0000A38A 72C4 <1> jc      short loc_mkdir_return
1634 <1>
1635 <1> loc_mkdir_save_added_cluster:
1636 0000A38C A1[5C430100] <1> mov     eax, [mkdir_LastDirCluster]
1637 0000A391 8B0D[58430100] <1> mov     ecx, [mkdir_FFCluster]
1638 <1> ; 01/03/2016
1639 0000A397 31D2 <1> xor     edx, edx
1640 0000A399 8915[06410100] <1> mov     [FAT_ClusterCounter], edx ; 0 ; reset
1641 0000A39F E803180000 <1> call    update_cluster
1642 0000A3A4 7304 <1> jnc     short loc_mkdir_save_fat_buffer_0
1643 0000A3A6 09C0 <1> or      eax, eax ; EAX = 0 -> cluster value is 0 or eocc
1644 0000A3A8 7518 <1> jnz     short loc_mkdir_save_fat_buffer_stc_retn
1645 <1>
1646 <1> loc_mkdir_save_fat_buffer_0:
1647 0000A3AA A1[58430100] <1> mov     eax, [mkdir_FFCluster]
1648 0000A3AF A3[5C430100] <1> mov     [mkdir_LastDirCluster], eax
1649 <1>
1650 0000A3B4 31C9 <1> xor     ecx, ecx
1651 0000A3B6 49 <1> dec     ecx ; FFFFFFFFh
1652 <1> ; ESI = Logical DOS Drive Description Table address
1653 0000A3B7 E8EB170000 <1> call    update_cluster
1654 0000A3BC 731A <1> jnc     short loc_mkdir_save_fat_buffer_1
1655 0000A3BE 09C0 <1> or      eax, eax
1656 0000A3C0 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 0000A3C2 803D[06410100]01 <1> cmp     byte [FAT_ClusterCounter], 1
1661 0000A3C9 720C <1> jb      short loc_mkdir_save_fat_buffer_retn
1662 <1>
1663 0000A3CB 66BB00FF <1> mov     bx, 0FF00h ; recalculate free space (BL = 0)
1664 <1> ; (BH = FFh -> Use ESI as Drv Param. Tbl.)
1665 0000A3CF 50 <1> push    eax
1666 0000A3D0 E8241B0000 <1> call    calculate_fat_freespace
1667 0000A3D5 58 <1> pop     eax
1668 0000A3D6 F9 <1> stc
1669 <1> loc_mkdir_save_fat_buffer_retn:
1670 0000A3D7 C3 <1> retn
1671 <1>
1672 <1> loc_mkdir_save_fat_buffer_1:
1673 <1> ; byte [FAT_BuffValidData] = 2
1674 0000A3D8 E8871A0000 <1> call    save_fat_buffer
1675 0000A3DD 72E3 <1> jc      short loc_mkdir_save_fat_buffer_stc_retn
1676 <1>
1677 <1> ; 01/03/2016
1678 0000A3DF 803D[06410100]01 <1> cmp     byte [FAT_ClusterCounter], 1
1679 0000A3E6 721B <1> jb      short loc_mkdir_save_fat_buffer_2
1680 <1>
1681 <1> ; ESI = Logical DOS Drive Description Table address
1682 0000A3E8 A1[06410100] <1> mov     eax, [FAT_ClusterCounter]
1683 0000A3ED 66BB01FF <1> mov     bx, 0FF01h ; add free clusters
1684 0000A3F1 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 0000A3F6 09C9 <1> or      ecx, ecx
1691 0000A3F8 7409 <1> jz      short loc_mkdir_save_fat_buffer_2
1692 <1>
1693 0000A3FA 66BB00FF <1> mov     bx, 0FF00h ; ; recalculate free space
1694 0000A3FE E8F61A0000 <1> call    calculate_fat_freespace
1695 <1>
1696 <1> loc_mkdir_save_fat_buffer_2:
1697 0000A403 C605[67430100]01 <1> mov     byte [mkdir_add_new_cluster], 1
1698 0000A40A E9C4000000 <1> jmp     loc_mkdir_upd_parent_dir_lmdt
1699 <1>
1700 <1> loc_mkdir_update_sub_dir_cluster:
1701 0000A40F A1[58430100] <1> mov     eax, [mkdir_FFCluster]
1702 0000A414 29C9 <1> sub     ecx, ecx ; 0
1703 <1> ; 01/03/2016
1704 0000A416 890D[06410100] <1> mov     [FAT_ClusterCounter], ecx ; 0 ; Reset
1705 0000A41C 49 <1> dec     ecx ; 0FFFFFFFFh
1706 <1>
1707 <1> ; ESI = Logical DOS Drive Descisption Table address
1708 0000A41D E885170000 <1> call    update_cluster
1709 0000A422 7379 <1> jnc     short loc_mkdir_save_fat_buffer_3
1710 0000A424 09C0 <1> or      eax, eax ; EAX = 0 -> cluster value is 0 or eocc

```



```

1711 0000A426 7475      <1>      jz      short loc_mkdir_save_fat_buffer_3
1712                   <1>      ; 01/03/2016
1713 0000A428 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 0000A42A A1[58430100] <1>      mov     eax, [mkdir_FFCluster]
1718                   <1>      ; Load disk sectors as a directory cluster
1719 0000A42F E82F160000 <1>      call    load_FAT_sub_directory
1720 0000A434 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 0000A436 BF40000800 <1>      mov     edi, Directory_Buffer + 64 ; 26/02/2016
1726                   <1>
1727                   <1>      ; 02/03/2016
1728 0000A43B 668B4611 <1>      mov     ax, [esi+LD_BPB+BytesPerSec] ; 512
1729 0000A43F 66C1E802 <1>      shr     ax, 2 ; 'byte count / 4' for 'stosd'
1730 0000A443 F7E1      <1>      mul     ecx
1731 0000A445 89C1      <1>      mov     ecx, eax
1732 0000A447 6629C0 <1>      sub     ax, ax
1733 0000A44A 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 0000A44C BF00000800 <1>      mov     edi, Directory_Buffer ; 26/02/2016
1744                   <1>
1745 0000A451 56         <1>      push    esi
1746                   <1>
1747 0000A452 BE[68430100] <1>      mov     esi, mkdir_Name
1748 0000A457 66C7062E00 <1>      mov     word [esi], 2Eh ; db '.', '0'
1749                   <1>
1750 0000A45C A1[58430100] <1>      mov     eax, [mkdir_FFCluster]
1751 0000A461 66B91000 <1>      mov     cx, 10h ; CL = Directory attribute
1752                   <1>      ; CH = 0 -> File size is 0
1753 0000A465 E883000000 <1>      call    make_directory_entry
1754                   <1>
1755 0000A46A 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 0000A46F 29C0      <1>      sub     eax, eax
1771 0000A471 3805[E4380100] <1>      cmp     byte [Current_Dir_Level], al ; 0
1772 0000A477 7605      <1>      jna     short loc_mkdir_set_ff_dir_entry_3
1773 0000A479 A1[E0380100] <1>      mov     eax, [Current_Dir_FCluster] ; parent dir
1774                   <1> loc_mkdir_set_ff_dir_entry_3:
1775 0000A47E 66C746012E00 <1>      mov     word [esi+1], 2Eh ; db '.', '0'
1776                   <1>
1777                   <1>      ;mov cx, 10h
1778 0000A484 E864000000 <1>      call    make_directory_entry
1779                   <1>
1780 0000A489 5E         <1>      pop     esi
1781                   <1>
1782 0000A48A C605[10410100]02 <1>      mov     byte [DirBuff_ValidData], 2
1783 0000A491 E834010000 <1>      call    save_directory_buffer
1784 0000A496 0F8373FFFFFF <1>      jnc     loc_mkdir_update_sub_dir_cluster
1785                   <1>
1786                   <1> retn_make_fat_directory:
1787 0000A49C 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 0000A49D E8C2190000 <1>      call    save_fat_buffer
1793 0000A4A2 0F821AFFFFFF <1>      jc      loc_mkdir_save_fat_buffer_stc_retn
1794                   <1>
1795 0000A4A8 803D[06410100]01 <1>      cmp     byte [FAT_ClusterCounter], 1
1796 0000A4AF 721B      <1>      jb      short loc_mkdir_save_fat_buffer_4
1797                   <1>
1798                   <1>      ; ESI = Logical DOS Drive Description Table address
1799 0000A4B1 A1[06410100] <1>      mov     eax, [FAT_ClusterCounter]
1800 0000A4B6 66BB01FF <1>      mov     bx, 0FF01h ; add free clusters
1801 0000A4BA E83A1A0000 <1>      call    calculate_fat_freespace
1802                   <1>
1803                   <1>      ;inc eax ; 0FFFFFFFh -> 0 ; recalculation is needed!
1804                   <1>      ;jnz short loc_mkdir_save_fat_buffer_4
1805                   <1>
1806                   <1>      ; ecx > 0 -> Recalculation is needed
1807 0000A4BF 09C9      <1>      or      ecx, ecx
1808 0000A4C1 7409      <1>      jz      short loc_mkdir_save_fat_buffer_4
1809                   <1>
1810 0000A4C3 66BB00FF <1>      mov     bx, 0FF00h ; recalculate free space
1811 0000A4C7 E82D1A0000 <1>      call    calculate_fat_freespace
1812                   <1>

```

```

1813 <1> loc_mkdir_save_fat_buffer_4:
1814 0000A4CC C605[67430100]00 <1> mov byte [mkdir_add_new_cluster], 0
1815 <1>
1816 <1> loc_mkdir_upd_parent_dir_lmdt:
1817 0000A4D3 E88D010000 <1> call update_parent_dir_lmdt
1818 <1>
1819 <1> ; 01/03/2016
1820 0000A4D8 803D[67430100]00 <1> cmp byte [mkdir_add_new_cluster], 0
1821 0000A4DF 0F8723FEFFFF <1> ja loc_mkdir_gffc_2
1822 <1>
1823 <1> loc_mkdir_retn_new_dir_cluster:
1824 0000A4E5 A1[58430100] <1> mov eax, [mkdir_FFCluster]
1825 0000A4EA 31D2 <1> xor edx, edx
1826 <1> loc_mkdir_retn:
1827 0000A4EC 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 0000A4ED 51 <1> push ecx
1852 <1>
1853 0000A4EE 884F0B <1> mov [edi+11], cl ; Attributes
1854 0000A4F1 6689471A <1> mov [edi+26], ax ; FClusterLw, 26
1855 0000A4F5 C1E810 <1> shr eax, 16
1856 0000A4F8 66894714 <1> mov [edi+20], ax ; FClusterHw, 20
1857 0000A4FC 6631C0 <1> xor ax, ax
1858 0000A4FF 6689470C <1> mov [edi+12], ax ; NTReserved, 12
1859 <1> ; CrtTimeTenth, 13
1860 0000A503 08ED <1> or ch, ch
1861 0000A505 7402 <1> jz short loc_make_direntry_set_filesize
1862 <1>
1863 0000A507 8B03 <1> mov eax, [ebx]
1864 <1>
1865 <1> loc_make_direntry_set_filesize:
1866 0000A509 89471C <1> mov [edi+28], eax ; FileSize, 28
1867 <1>
1868 0000A50C 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 0000A511 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 0000A516 6689470E <1> mov [edi+14], ax ; CrtTime, 14
1876 0000A51A 66895710 <1> mov [edi+16], dx ; CrtDate, 16
1877 0000A51E 66895712 <1> mov [edi+18], dx ; LastAccDate, 18
1878 0000A522 66894716 <1> mov [edi+22], ax ; WrtTime, 14
1879 0000A526 66895718 <1> mov [edi+24], dx ; WrtDate, 16
1880 0000A52A 59 <1> pop ecx
1881 <1>
1882 0000A52B 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 0000A52C B404 <1> mov ah, 04h ; Return Current Date
1893 0000A52E E8A0B3FFFF <1> call int1Ah
1894 <1>
1895 0000A533 88E8 <1> mov al, ch ; <- century BCD
1896 0000A535 240F <1> and al, 0Fh
1897 0000A537 88EC <1> mov ah, ch
1898 0000A539 C0EC04 <1> shr ah, 4
1899 0000A53C D50A <1> aad
1900 0000A53E 88C5 <1> mov ch, al ; -> century
1901 <1>
1902 0000A540 88C8 <1> mov al, cl ; <- year BCD
1903 0000A542 240F <1> and al, 0Fh
1904 0000A544 88CC <1> mov ah, cl
1905 0000A546 C0EC04 <1> shr ah, 4
1906 0000A549 D50A <1> aad
1907 0000A54B 88C1 <1> mov cl, al ; -> year
1908 <1>
1909 0000A54D 88E8 <1> mov al, ch
1910 0000A54F B464 <1> mov ah, 100
1911 0000A551 F6E4 <1> mul ah
1912 0000A553 30ED <1> xor ch, ch
1913 0000A555 6601C8 <1> add ax, cx
1914 0000A558 662DBC07 <1> sub ax, 1980 ; ms-dos epoch

```

```

1915 0000A55C 6689C1      <1>      mov     cx, ax
1916                      <1>
1917 0000A55F 88F0        <1>      mov     al, dh ; <- month in bcd
1918 0000A561 240F        <1>      and     al, 0Fh
1919 0000A563 88F4        <1>      mov     ah, dh
1920 0000A565 C0EC04      <1>      shr     ah, 4
1921 0000A568 D50A        <1>      aad
1922 0000A56A 88C6        <1>      mov     dh, al ; -> month
1923                      <1>
1924 0000A56C 88D0        <1>      mov     al, dl ; <- day BCD
1925 0000A56E 240F        <1>      and     al, 0Fh
1926 0000A570 88D4        <1>      mov     ah, dl
1927 0000A572 C0EC04      <1>      shr     ah, 4
1928 0000A575 D50A        <1>      aad
1929 0000A577 88C2        <1>      mov     dl, al ; -> day
1930                      <1>
1931 0000A579 88C8        <1>      mov     al, cl ; count of years from 1980
1932 0000A57B 66C1E004     <1>      shl     ax, 4
1933 0000A57F 08F0        <1>      or      al, dh ; month of year, 1 to 12
1934 0000A581 66C1E005     <1>      shl     ax, 5
1935 0000A585 08D0        <1>      or      al, dl ; day of year, 1 to 31
1936                      <1>
1937 0000A587 6650        <1>      push    ax ; push date
1938                      <1>
1939 0000A589 B402        <1>      mov     ah, 02h ; Return Current Time
1940 0000A58B E843B3FFFF     <1>      call    int1Ah
1941                      <1>
1942 0000A590 88E8        <1>      mov     al, ch ; <- hours BCD
1943 0000A592 240F        <1>      and     al, 0Fh
1944 0000A594 88EC        <1>      mov     ah, ch
1945 0000A596 C0EC04      <1>      shr     ah, 4
1946 0000A599 D50A        <1>      aad
1947 0000A59B 88C5        <1>      mov     ch, al ; -> hours
1948                      <1>
1949 0000A59D 88C8        <1>      mov     al, cl ; <- minutes BCD
1950 0000A59F 240F        <1>      and     al, 0Fh
1951 0000A5A1 88CC        <1>      mov     ah, cl
1952 0000A5A3 C0EC04      <1>      shr     ah, 4
1953 0000A5A6 D50A        <1>      aad
1954 0000A5A8 88C1        <1>      mov     cl, al ; -> minutes
1955                      <1>
1956 0000A5AA 88F0        <1>      mov     al, dh ; <- seconds BCD
1957 0000A5AC 240F        <1>      and     al, 0Fh
1958 0000A5AE 88F4        <1>      mov     ah, dh
1959 0000A5B0 C0EC04      <1>      shr     ah, 4
1960 0000A5B3 D50A        <1>      aad
1961 0000A5B5 88C6        <1>      mov     dh, al ; -> seconds
1962                      <1>
1963 0000A5B7 88E8        <1>      mov     al, ch ; hours
1964 0000A5B9 66C1E006     <1>      shl     ax, 6
1965 0000A5BD 08C8        <1>      or      al, cl ; minutes
1966 0000A5BF 66C1E005     <1>      shl     ax, 5
1967 0000A5C3 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 0000A5C5 08F0        <1>      or      al, dh ; seconds
1971                      <1>
1972 0000A5C7 665A        <1>      pop     dx ; pop date
1973                      <1>
1974 0000A5C9 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 0000A5CA BB00000800     <1>      mov     ebx, Directory_Buffer
1994 0000A5CF 803D[10410100]02 <1>      cmp     byte [DirBuff_ValidData], 2
1995 0000A5D6 7403        <1>      je      short loc_save_dir_buffer
1996 0000A5D8 31C0        <1>      xor     eax, eax
1997 0000A5DA C3          <1>      retn
1998                      <1>
1999                      <1> loc_save_dir_buffer:
2000 0000A5DB 56          <1>      push    esi
2001 0000A5DC 31DB        <1>      xor     ebx, ebx
2002 0000A5DE 8A3D[0E410100] <1>      mov     bh, [DirBuff_DRV]
2003 0000A5E4 80EF41      <1>      sub     bh, 'A'
2004 0000A5E7 BE00010900     <1>      mov     esi, Logical_DOSDisks
2005 0000A5EC 01DE        <1>      add     esi, ebx
2006 0000A5EE 668B4E03     <1>      mov     cx, [esi+LD_FATType]
2007                      <1>      ; CH = FS Type (Alh for FS)
2008                      <1>      ; CL = FAT Type (0 for FS)
2009 0000A5F2 08C9        <1>      or      cl, cl
2010 0000A5F4 7433        <1>      jz      short loc_save_dir_buff_stc_retn
2011                      <1>
2012                      <1> loc_save_dir_buffer_check_cluster_no:
2013 0000A5F6 A1[15410100] <1>      mov     eax, [DirBuff_Cluster]
2014 0000A5FB 28FF        <1>      sub     bh, bh ; ebx = 0
2015 0000A5FD 09C0        <1>      or      eax, eax
2016 0000A5FF 7540        <1>      jnz     short loc_save_sub_dir_buffer

```

```

2017 0000A601 8A25[0F410100] <1> mov ah, [DirBuff_FATType]
2018 0000A607 FEC3 <1> inc bl ; bl = 1
2019 0000A609 38DC <1> cmp ah, bl
2020 0000A60B 721D <1> jb short loc_save_dir_buff_inv_data_retn
2021 0000A60D FEC3 <1> inc bl ; bl = 2
2022 0000A60F 38E3 <1> cmp bl, ah
2023 0000A611 7217 <1> jb short loc_save_dir_buff_inv_data_retn
2024 <1>
2025 <1> loc_save_root_dir_buffer:
2026 0000A613 668B5E17 <1> mov bx, [esi+LD_BPB+RootDirEnts]
2027 0000A617 6683C30F <1> add bx, 15
2028 0000A61B 66C1EB04 <1> shr bx, 4 ; 16 dir entries per sector
2029 0000A61F 6609DB <1> or bx, bx
2030 0000A622 7405 <1> jz short loc_save_dir_buff_stc_retn
2031 <1> ;mov ecx, ebx
2032 0000A624 8B4664 <1> mov eax, [esi+LD_ROOTBegin] ; 26/02/2016
2033 0000A627 EB23 <1> jmp short loc_write_directory_to_disk
2034 <1>
2035 <1> loc_save_dir_buff_stc_retn:
2036 0000A629 F9 <1> stc
2037 <1> loc_save_dir_buff_inv_data_retn:
2038 <1> ; 15/10/2016 (0Dh -> 29)
2039 0000A62A B01D <1> mov al, 29 ; Invalid data !
2040 0000A62C C605[10410100]00 <1> mov byte [DirBuff_ValidData], 0
2041 0000A633 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 0000A635 B812000000 <1> mov eax, 18 ; Drive not ready or write error
2046 <1>
2047 <1> loc_save_dir_buff_retn:
2048 0000A63A BB00000800 <1> mov ebx, Directory_Buffer
2049 0000A63F 5E <1> pop esi
2050 0000A640 C3 <1> retn
2051 <1>
2052 <1> loc_save_sub_dir_buffer:
2053 <1> ; ebx = 0
2054 0000A641 83E802 <1> sub eax, 2
2055 0000A644 8A5E13 <1> mov bl, [esi+LD_BPB+SecPerClust]
2056 0000A647 F7E3 <1> mul ebx
2057 0000A649 034668 <1> add eax, [esi+LD_DATABegin]
2058 <1> ;mov ecx, ebx
2059 <1>
2060 <1> loc_write_directory_to_disk:
2061 0000A64C 89D9 <1> mov ecx, ebx
2062 0000A64E BB00000800 <1> mov ebx, Directory_Buffer
2063 0000A653 E8704B0000 <1> call disk_write
2064 0000A658 72DB <1> jc short loc_write_directory_to_disk_err
2065 <1>
2066 <1> loc_save_dir_buff_validate_retn:
2067 0000A65A C605[10410100]01 <1> mov byte [DirBuff_ValidData], 1
2068 0000A661 31C0 <1> xor eax, eax
2069 <1> ; 26/02/2016
2070 0000A663 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 0000A665 29C0 <1> sub eax, eax
2086 0000A667 8A25[E4380100] <1> mov ah, [Current_Dir_Level]
2087 0000A66D A0[E5380100] <1> mov al, [Current_FATType]
2088 0000A672 3C01 <1> cmp al, 1
2089 0000A674 723A <1> jb short loc_UPDLMDT_proc_retn
2090 <1>
2091 <1> loc_update_parent_dir_lm_date_time:
2092 0000A676 08E4 <1> or ah, ah
2093 0000A678 7436 <1> jz short loc_UPDLMDT_proc_retn
2094 <1>
2095 0000A67A 56 <1> push esi ; *
2096 0000A67B 8825[88430100] <1> mov [UPDLMDT_CDirLevel], ah
2097 0000A681 8B15[E0380100] <1> mov edx, [Current_Dir_FCluster]
2098 0000A687 8915[89430100] <1> mov [UPDLMDT_CDirFCluster], edx
2099 <1>
2100 0000A68D FECC <1> dec ah
2101 0000A68F B90C000000 <1> mov ecx, 12
2102 0000A694 BE[47410100] <1> mov esi, PATH_Array
2103 <1>
2104 0000A699 8825[E4380100] <1> mov [Current_Dir_Level], ah
2105 0000A69F 08E4 <1> or ah, ah
2106 0000A6A1 750E <1> jnz short loc_update_parent_dir_lmdt_load_sub_dir_1
2107 0000A6A3 803D[E5380100]02 <1> cmp byte [Current_FATType], 2
2108 0000A6AA 770B <1> ja short loc_update_parent_dir_lmdt_load_sub_dir_2
2109 0000A6AC 28C0 <1> sub al, al ; eax = 0
2110 0000A6AE EB0A <1> jmp short loc_update_parent_dir_lmdt_load_sub_dir_3
2111 <1>
2112 <1> loc_UPDLMDT_proc_retn:
2113 0000A6B0 C3 <1> retn
2114 <1>
2115 <1> loc_update_parent_dir_lmdt_load_sub_dir_1:
2116 0000A6B1 B010 <1> mov al, 16
2117 0000A6B3 F6E4 <1> mul ah
2118 0000A6B5 01C6 <1> add esi, eax

```



```

2119                                     <1>
2120                                     <1> loc_update_parent_dir_lmdt_load_sub_dir_2:
2121 0000A6B7 8B460C                     <1>     mov     eax, [esi+12] ; Parent Dir First Cluster
2122                                     <1>
2123                                     <1> loc_update_parent_dir_lmdt_load_sub_dir_3:
2124 0000A6BA A3[E0380100]             <1>     mov     [Current_Dir_FCluster], eax
2125                                     <1>
2126 0000A6BF 83C610                     <1>     add     esi, 16
2127 0000A6C2 66BF[6E42]               <1>     mov     di, Dir_File_Name
2128 0000A6C6 F3A4                     <1>     rep     movsb
2129                                     <1>
2130 0000A6C8 BE00010900                <1>     mov     esi, Logical_DOSDisks
2131 0000A6CD 29DB                     <1>     sub     ebx, ebx
2132 0000A6CF 8A3D[E6380100]           <1>     mov     bh, [Current_Drv]
2133 0000A6D5 01DE                     <1>     add     esi, ebx
2134 0000A6D7 E88FF7FFFF               <1>     call    reload_current_directory
2135 0000A6DC 7232                     <1>     jc      short loc_update_parent_dir_lmdt_restore_cdirlevel
2136                                     <1>
2137                                     <1> loc_update_parent_dir_lmdt_locate_dir:
2138 0000A6DE BE[6E420100]             <1>     mov     esi, Dir_File_Name
2139 0000A6E3 6631C9                   <1>     xor     cx, cx
2140 0000A6E6 66B81008                 <1>     mov     ax, 0810h ; Only directories
2141 0000A6EA E8B7F6FFFF               <1>     call    locate_current_dir_file
2142                                     <1> ; EDI = DirBuff Directory Entry Address
2143 0000A6EF 721F                     <1>     jc      short loc_update_parent_dir_lmdt_restore_cdirlevel
2144                                     <1>
2145 0000A6F1 E836FEFFFF               <1>     call    convert_current_date_time
2146 0000A6F6 66895712                 <1>     mov     [edi+18], dx ; Last Access Date
2147 0000A6FA 66895718                 <1>     mov     [edi+24], dx ; Last Write Date
2148 0000A6FE 66894716                 <1>     mov     [edi+22], ax ; Last Write Time
2149                                     <1>
2150 0000A702 C605[10410100]02         <1>     mov     byte [DirBuff_ValidData], 2
2151 0000A709 E8BCFEFFFF               <1>     call    save_directory_buffer
2152 0000A70E 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 0000A710 8A25[88430100]           <1>     mov     ah, [UPDLMDT_CDirLevel]
2157 0000A716 8825[E4380100]           <1>     mov     [Current_Dir_Level], ah
2158 0000A71C 8B15[89430100]           <1>     mov     edx, [UPDLMDT_CDirFCluster]
2159 0000A722 8915[E0380100]           <1>     mov     [Current_Dir_FCluster], edx
2160                                     <1>
2161 0000A728 5E                       <1>     pop     esi ; *
2162 0000A729 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')
2167                                     <1> ; 14/03/2010
2168                                     <1> ; INPUT ->
2169                                     <1> ; EAX = Directory Entry (Index) Number (< 65536)
2170                                     <1> ; OUTPUT ->
2171                                     <1> ; cf = 0 -> OK (EAX = 0)
2172                                     <1> ; cf = 1 -> error code in EAX (AL)
2173                                     <1> ;
2174                                     <1> ; (Modified registers: EAX, EDX, ECX, EBX, EDI)
2175                                     <1>
2176 0000A72A 66A3[B8430100]           <1>     mov     [DLN_EntryNumber], ax
2177 0000A730 C605[BA430100]40         <1>     mov     byte [DLN_40h], 40h
2178                                     <1>
2179 0000A737 E858000000               <1>     call    locate_current_dir_entry
2180 0000A73C 7308                     <1>     jnc     short loc_dln_check_attributes
2181 0000A73E C3                       <1>     retn
2182                                     <1>
2183                                     <1> loc_dln_longname_not_found:
2184 0000A73F B802000000               <1>     mov     eax, 2
2185 0000A744 F9                       <1>     stc
2186 0000A745 C3                       <1>     retn
2187                                     <1>
2188                                     <1> loc_dln_check_attributes:
2189 0000A746 B00F                     <1>     mov     al, 0Fh ; long name
2190 0000A748 8A670B                   <1>     mov     ah, [edi+0Bh] ; dir entry attributes
2191 0000A74B 38C4                     <1>     cmp     ah, al
2192 0000A74D 75F0                     <1>     jne     short loc_dln_longname_not_found
2193 0000A74F 8A27                     <1>     mov     ah, [edi]
2194 0000A751 2A25[BA430100]           <1>     sub     ah, [DLN_40h]
2195 0000A757 76E6                     <1>     jna     short loc_dln_longname_not_found
2196 0000A759 80FC14                   <1>     cmp     ah, 14h ; 84-64=20 -> 20*13=260 bytes
2197 0000A75C 77E1                     <1>     ja      short loc_dln_longname_not_found
2198                                     <1>
2199 0000A75E C607E5                   <1>     mov     byte [edi], 0E5h ; deleted sign
2200 0000A761 C605[10410100]02         <1>     mov     byte [DirBuff_ValidData], 2 ; changed/write sign
2201 0000A768 C605[BA430100]00         <1>     mov     byte [DLN_40h], 0 ; 40h -> 0
2202                                     <1>
2203                                     <1> loc_dln_delete_next_ln_entry:
2204 0000A76F 80FC01                   <1>     cmp     ah, 1
2205 0000A772 7616                     <1>     jna     short loc_dln_longname_retn
2206                                     <1> loc_dln_delete_next_ln_entry_0:
2207 0000A774 66FF05[B8430100]         <1>     inc     word [DLN_EntryNumber]
2208 0000A77B 0FB705[B8430100]         <1>     movzx   eax, word [DLN_EntryNumber]
2209 0000A782 E80D000000               <1>     call    locate_current_dir_entry
2210 0000A787 73BD                     <1>     jnc     short loc_dln_check_attributes
2211                                     <1>
2212                                     <1> loc_dln_longname_stc_retn:
2213 0000A789 C3                       <1>     retn
2214                                     <1>
2215                                     <1> loc_dln_longname_retn:
2216                                     <1> ;cmp byte [DirBuff_ValidData], 2
2217                                     <1> ;jne short loc_dln_longname_retn_xor_eax
2218 0000A78A E83BFEFFFF               <1>     call    save_directory_buffer
2219 0000A78F 72F8                     <1>     jc      short loc_dln_longname_stc_retn
2220                                     <1>

```

```

2221 <1> loc_dln_longname_retn_xor_eax:
2222 <1>     xor     eax, eax
2223 <1>     retn
2224 <1>
2225 <1> locate_current_dir_entry:
2226 <1>     ; 16/10/2016
2227 <1>     ; 15/10/2016
2228 <1>     ; 23/03/2016
2229 <1>     ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)
2230 <1>     ; 01/08/2011 (DIR.ASM, 'proc_locate_current_dir_entry')
2231 <1>     ; 07/03/2010
2232 <1>     ; INPUT ->
2233 <1>     ;     EAX = Directory Entry (Index) Number (< 65536)
2234 <1>     ; OUTPUT ->
2235 <1>     ;     EDI = Directory Entry Address
2236 <1>     ;     EAX = Cluster Number of Directory Buffer
2237 <1>     ;     EBX = Directory Buffer Entry Offset
2238 <1>     ;     ECX = DirBuff Valid Data identifier (CL)
2239 <1>     ;     If CF = 0 and CL = 2 then
2240 <1>     ;         directory buffer modified and
2241 <1>     ;         must be written to disk.
2242 <1>     ;     If CF = 0 and CL = 1 then
2243 <1>     ;         dir buffer has been written to disk, already.
2244 <1>     ;     CF = 1 -> Error code in EAX (AL)
2245 <1>     ;
2246 <1>     ; (Modified registers: EAX, EDX, ECX, EBX, EDI)
2247 <1>
2248 <1> loc_locate_current_dir_entry:
2249 <1>     push    esi
2250 <1>     mov     ecx, eax
2251 <1>     mov     edx, 32
2252 <1>     mul     edx
2253 <1>     mov     [LCDE_ByteOffset], eax
2254 <1>     xor     ebx, ebx
2255 <1>     mov     bh, [Current_Drv]
2256 <1>     mov     al, [DirBuff_Drv]
2257 <1>     sub     al, 'A'
2258 <1>     mov     esi, Logical_DOSDisks
2259 <1>     add     esi, ebx
2260 <1>     cmp     bh, al
2261 <1>     jne     loc_lcde_reload_current_directory
2262 <1> loc_lcde_cdl_check:
2263 <1>     cmp     byte [Current_Dir_Level], 0
2264 <1>     ja      short loc_lcde_calc_dirbuff_cluster_offset
2265 <1>     ; 27/02/2016
2266 <1>     ; TRDOS v1 has bug here for FAT32 fs !
2267 <1>     ; (Root Directory Entries for FAT32 = 0)
2268 <1>     cmp     byte [esi+LD_FATType], 3 ; FAT32
2269 <1>     jnb     short loc_lcde_calc_dirbuff_cluster_offset
2270 <1>
2271 <1> loc_lcde_cdl_check_FAT12_16:
2272 <1>     mov     ax, [esi+LD_BPB+RootDirEnts]
2273 <1>     dec     ax
2274 <1>     ;xor     dx, dx
2275 <1>     cmp     ax, cx ; cx = Directory Entry (Index) Number
2276 <1>     jb      short loc_lcde_stc_12h_retn
2277 <1>     mov     [LCDE_EntryIndex], cx
2278 <1>     xor     eax, eax
2279 <1>     jmp     loc_lcde_check_dir_buffer_cluster
2280 <1>
2281 <1> loc_lcde_stc_12h_retn:
2282 <1>     pop     esi
2283 <1>     mov     ebx, ecx
2284 <1>     mov     ecx, edx
2285 <1>     ; 16/10/2016 (12h -> 12)
2286 <1>     mov     eax, 12 ; No more files
2287 <1>     retn
2288 <1>
2289 <1> loc_lcde_calc_dirbuff_cluster_offset:
2290 <1>     mov     bl, [esi+LD_BPB+SecPerClust]
2291 <1>     xor     bh, bh
2292 <1>     mov     ax, [esi+LD_BPB+BytesPerSec]
2293 <1>     mul     bx
2294 <1>     or      dx, dx ; If bytes per cluster > 32KB it is invalid
2295 <1>     jnz     short loc_lcde_invalid_format
2296 <1>     ;mov     ecx, eax
2297 <1>     mov     cx, ax ; BYTES PER CLUSTER
2298 <1>     mov     eax, [LCDE_ByteOffset]
2299 <1>     ;sub     edx, edx
2300 <1>     div     ecx
2301 <1>     cmp     eax, 65535
2302 <1>     ja      short loc_lcde_invalid_format
2303 <1>
2304 <1>     ; cluster sequence number of directory (< 65536)
2305 <1>     mov     [LCDE_ClusterSN], ax
2306 <1>
2307 <1>     mov     ax, dx ; byte offset in cluster (directory buffer)
2308 <1>     mov     bx, 32 ; ; 1 dir entry = 32 bytes
2309 <1>     sub     dx, dx ; 0
2310 <1>     div     bx
2311 <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 <1>     mov     eax, [Current_Dir_FCluster]
2315 <1>
2316 <1> loc_lcde_get_next_cluster:
2317 <1>     cmp     word [LCDE_ClusterSN], 0
2318 <1>     jna     short loc_lcde_check_dir_buffer_cluster
2319 <1>     mov     [LCDE_Cluster], eax
2320 <1>     call    get_next_cluster
2321 <1>     jc      short loc_lcde_check_gnc_error
2322 <1>     dec     word [LCDE_ClusterSN]

```

```

2323 0000A851 EBE1      <1>      jmp      short loc_lcde_get_next_cluster
2324                  <1>
2325                  <1> loc_lcde_reload_current_directory:
2326 0000A853 51        <1>      push    ecx
2327 0000A854 E812F6FFFF <1>      call    reload_current_directory
2328 0000A859 59        <1>      pop     ecx
2329 0000A85A 0F8361FFFF <1>      jnc     loc_lcde_cdl_check
2330 0000A860 5E        <1>      pop     esi
2331 0000A861 C3        <1>      retn
2332                  <1>
2333                  <1> loc_lcde_invalid_format:
2334                  <1>      ; 15/10/2016 (0Bh -> 28)
2335 0000A862 B81C000000 <1>      mov     eax, 28 ; Invalid Format !
2336                  <1> loc_lcde_drive_not_ready_read_err:
2337 0000A867 F9        <1>      stc
2338 0000A868 5E        <1>      pop     esi
2339 0000A869 C3        <1>      retn
2340                  <1>
2341                  <1> loc_lcde_check_gnc_error:
2342 0000A86A 09C0      <1>      or      eax, eax
2343 0000A86C 75F9      <1>      jnz     short loc_lcde_drive_not_ready_read_err
2344 0000A86E 66FF0D[BE430100] <1>      dec     word [LCDE_ClusterSN]
2345 0000A875 75EB      <1>      jnz     short loc_lcde_invalid_format
2346 0000A877 A1[C0430100] <1>      mov     eax, [LCDE_Cluster]
2347                  <1>
2348                  <1> loc_lcde_check_dir_buffer_cluster:
2349 0000A87C 3B05[15410100] <1>      cmp     eax, [DirBuff_Cluster]
2350 0000A882 755C      <1>      jne     short loc_lcde_load_dir_cluster
2351 0000A884 803D[10410100]00 <1>      cmp     byte [DirBuff_ValidData], 0
2352 0000A88B 7727      <1>      ja      short loc_lcde_check_dir_buffer_cluster_next
2353 0000A88D 803D[E4380100]00 <1>      cmp     byte [Current_Dir_Level], 0
2354 0000A894 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 0000A896 807E0303 <1>      cmp     byte [esi+LD_FATType], 3 ; FAT32
2358 0000A89A 7359      <1>      jnb     short loc_lcde_load_dir_cluster_0
2359                  <1>      ;
2360 0000A89C 0FB74E17 <1>      movzx   ecx, word [esi+LD_BPB+RootDirEnts]
2361 0000A8A0 6683C10F <1>      add     cx, 15 ; round up (16 entries per sector)
2362 0000A8A4 66C1E904 <1>      shr     cx, 4 ; 1 sector contains 16 dir entries
2363                  <1>
2364 0000A8A8 8B4664 <1>      mov     eax, [esi+LD_ROOTBegin]
2365 0000A8AB EB54      <1>      jmp     short loc_lcde_load_dir_cluster_1
2366                  <1>
2367                  <1> loc_lcde_validate_dirBuff:
2368 0000A8AD C605[10410100]01 <1>      mov     byte [DirBuff_ValidData], 1
2369                  <1>
2370                  <1> loc_lcde_check_dir_buffer_cluster_next:
2371 0000A8B4 0FB71D[BC430100] <1>      movzx   ebx, word [LCDE_EntryIndex]
2372 0000A8BB 663B1D[13410100] <1>      cmp     bx, [DirBuff_LastEntry]
2373 0000A8C2 779E      <1>      ja      short loc_lcde_invalid_format
2374 0000A8C4 B820000000 <1>      mov     eax, 32
2375 0000A8C9 F7E3      <1>      mul     ebx
2376                  <1>      ;or     edx, edx
2377                  <1>      ;jnz     short loc_lcde_invalid_format
2378                  <1>
2379 0000A8CB BF00000800 <1>      mov     edi, Directory_Buffer
2380 0000A8D0 01C7      <1>      add     edi, eax ; add entry offset to buffer address
2381                  <1>
2382                  <1> loc_lcde_dir_buffer_last_check:
2383 0000A8D2 A1[15410100] <1>      mov     eax, [DirBuff_Cluster]
2384 0000A8D7 0FB60D[10410100] <1>      movzx   ecx, byte [DirBuff_ValidData]
2385                  <1>
2386                  <1> loc_lcde_retn:
2387 0000A8DE 5E        <1>      pop     esi
2388 0000A8DF 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 0000A8E0 50        <1>      push    eax
2394 0000A8E1 E8E4FCFFFF <1>      call    save_directory_buffer
2395 0000A8E6 58        <1>      pop     eax
2396 0000A8E7 72F5      <1>      jc      short loc_lcde_retn
2397                  <1>
2398                  <1> loc_lcde_load_dir_cluster_n2:
2399 0000A8E9 C605[10410100]00 <1>      mov     byte [DirBuff_ValidData], 0
2400 0000A8F0 A3[15410100] <1>      mov     [DirBuff_Cluster], eax
2401                  <1>
2402                  <1> loc_lcde_load_dir_cluster_0:
2403 0000A8F5 83E802 <1>      sub     eax, 2
2404 0000A8F8 0FB64E13 <1>      movzx   ecx, byte [esi+LD_BPB+SecPerClust]
2405 0000A8FC F7E1      <1>      mul     ecx
2406 0000A8FE 034668 <1>      add     eax, [esi+LD_DATABegin]
2407                  <1>
2408                  <1> loc_lcde_load_dir_cluster_1:
2409 0000A901 BB00000800 <1>      mov     ebx, Directory_Buffer
2410                  <1>      ; ecx = sector count
2411 0000A906 E8CC480000 <1>      call    disk_read
2412 0000A90B 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 0000A90D B811000000 <1>      mov     eax, 17 ; Drive not ready or read error !
2417 0000A912 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      0000A914 29C0      <1>      sub    eax, eax
2432      0000A916 88FC      <1>      mov    ah, bh
2433      0000A918 BE00010900 <1>      mov    esi, Logical_DOSDisks
2434      0000A91D 01C6      <1>      add    esi, eax
2435      <1>
2436      0000A91F 807E0301 <1>      cmp    byte [esi+LD_FATType], 1
2437      0000A923 7312      <1>      jnb    short loc_del_fat_file
2438      <1>
2439      0000A925 807E04A1 <1>      cmp    byte [esi+LD_FSType], 0A1h
2440      0000A929 7406      <1>      je     short loc_del_fs_file
2441      <1>
2442      <1> loc_del_file_invalid_format:
2443      0000A92B 30E4      <1>      xor    ah, ah
2444      <1>      ; 15/10/2016 (0Bh -> 28)
2445      0000A92D B01C      <1>      mov    al, 28 ; Invalid Format
2446      0000A92F F9       <1>      stc
2447      0000A930 C3       <1>      retn
2448      <1>
2449      <1> loc_del_fs_file:
2450      0000A931 E8400F0000 <1>      call   delete_fs_file
2451      0000A936 C3       <1>      retn
2452      <1>
2453      <1> loc_del_fat_file:
2454      0000A937 E808000000 <1>      call   delete_directory_entry
2455      0000A93C 7205      <1>      jc     short loc_del_file_err_retn
2456      <1>
2457      <1> loc_delfile_unlink_cluster_chain:
2458      0000A93E 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      0000A943 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      0000A944 881D[52430100] <1>      mov    [DelFile_LNEL], bl
2485      0000A94A 66890D[50430100] <1>      mov    [DelFile_EntryCounter], cx
2486      <1>
2487      0000A951 668B4714 <1>      mov    ax, [edi+20] ; First Cluster High Word
2488      0000A955 C1E010 <1>      shl    eax, 16
2489      0000A958 668B471A <1>      mov    ax, [edi+26] ; First Cluster Low Word
2490      <1>
2491      0000A95C A3[4C430100] <1>      mov    [DelFile_FCluster], eax
2492      <1>
2493      <1> loc_del_short_name:
2494      0000A961 C607E5 <1>      mov    byte [edi], 0E5h ; Deleted sign
2495      <1>
2496      0000A964 C605[10410100]02 <1>      mov    byte [DirBuff_ValidData], 2
2497      0000A96B E85AFCFFFF <1>      call   save_directory_buffer
2498      0000A970 723D <1>      jc     short loc_delete_direntry_err_return
2499      <1>
2500      <1> loc_del_long_name:
2501      0000A972 0FB615[52430100] <1>      movzx  edx, byte [DelFile_LNEL]
2502      0000A979 08D2 <1>      or     dl, dl
2503      0000A97B 7416 <1>      jz     short loc_del_dir_entry_update_parent_dir_lm_date
2504      <1>
2505      0000A97D 8835[52430100] <1>      mov    byte [DelFile_LNEL], dh ; 0
2506      <1>
2507      0000A983 0FB705[50430100] <1>      movzx  eax, word [DelFile_EntryCounter]
2508      0000A98A 29D0 <1>      sub    eax, edx
2509      <1>      ;jnc    short loc_del_long_name_continue
2510      0000A98C 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      0000A98E 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      0000A993 801D[52430100]00 <1>      sbb    byte [DelFile_LNEL], 0 ; 0FFh if cf = 1
2523      <1>
2524      0000A99A E8C6FCFFFF <1>      call   update_parent_dir_lmdt
2525      0000A99F B700 <1>      mov    bh, 0
2526      0000A9A1 80D700 <1>      adc    bh, 0

```



```

2527                                     <1>
2528 0000A9A4 8A1D[52430100]          <1>      mov     bl, byte [DelFile_LNEL]
2529                                     <1>
2530                                     <1> loc_delete_dirent:
2531 0000A9AA A1[4C430100]          <1>      mov     eax, [DelFile_FCluster]
2532                                     <1> loc_delete_dirent_err_return:
2533 0000A9AF 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>      ;     (ASCII 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 0000A9B0 803D[E5380100]00        <1>      cmp     byte [Current_FATType], 0
2553 0000A9B7 7706                    <1>      ja      short loc_rename_directory_entry
2554                                     <1>
2555 0000A9B9 E8B90E0000              <1>      call    rename_fs_file_or_directory
2556 0000A9BE C3                      <1>      retn
2557                                     <1>
2558                                     <1> loc_rename_directory_entry:
2559 0000A9BF 881D[52430100]          <1>      mov     [DelFile_LNEL], bl
2560 0000A9C5 66890D[50430100]        <1>      mov     [DelFile_EntryCounter], cx
2561 0000A9CC A3[4C430100]            <1>      mov     [DelFile_FCluster], eax
2562                                     <1>
2563 0000A9D1 0FB7C1                    <1>      movzx   eax, cx
2564 0000A9D4 E8BBFDFFFF              <1>      call    locate_current_dir_entry
2565 0000A9D9 7308                    <1>      jnc     short loc_rename_dirent_check_fcluster
2566                                     <1>
2567                                     <1> loc_rename_dirent_pop_retn:
2568 0000A9DB C3                      <1>      retn
2569                                     <1>
2570                                     <1> loc_rename_dirent_pop_invd_retn:
2571 0000A9DC F9                      <1>      stc
2572                                     <1> loc_rename_dirent_invd_retn:
2573                                     <1>      ; 15/10/2016 (0Dh -> 29)
2574 0000A9DD B81D000000              <1>      mov     eax, 29 ; Invalid data
2575                                     <1> loc_rename_retn:
2576 0000A9E2 C3                      <1>      retn
2577                                     <1>
2578                                     <1> loc_rename_dirent_check_fcluster:
2579 0000A9E3 668B5714                <1>      mov     dx, [edi+20] ; First Cluster HW
2580 0000A9E7 66C1E210                <1>      shl     dx, 16
2581 0000A9EB 668B571A                <1>      mov     dx, [edi+26] ; First Cluster LW
2582 0000A9EF 3B15[4C430100]          <1>      cmp     edx, [DelFile_FCluster]
2583 0000A9F5 75E5                    <1>      jne     short loc_rename_dirent_pop_invd_retn
2584                                     <1>      ; ESI = New file (or directory) name. (ASCII string)
2585                                     <1>      ; 06/03/2016
2586                                     <1>      ; TRDOS v2 - NOTE: 'convert_file_name' procedure
2587                                     <1>      ; has been modified for eliminating following situation.
2588                                     <1>      ;
2589                                     <1>      ; TRDOS v1 - NOTE: If file/dir name is more than 11 bytes
2590                                     <1>      ; without a dot, attributes (edi+11) byte will be overwritten !
2591                                     <1>      ; (Dot file name input must be proper for 11 byte dir entry
2592                                     <1>      ; type file name output.)
2593 0000A9F7 E89FF6FFFF              <1>      call    convert_file_name
2594                                     <1>
2595 0000A9FC C605[10410100]02        <1>      mov     byte [DirBuff_ValidData], 2
2596 0000AA03 E8C2FBFFFF              <1>      call    save_directory_buffer
2597 0000AA08 72D8                    <1>      jc      short loc_rename_retn
2598                                     <1>
2599                                     <1> loc_rename_dirent_del_ln:
2600 0000AA0A 0FB615[52430100]          <1>      movzx   edx, byte [DelFile_LNEL]
2601 0000AA11 08D2                    <1>      or      dl, dl
2602 0000AA13 7410                    <1>      jz      short loc_rename_dirent_update_parent_dir_lm_date
2603                                     <1>
2604 0000AA15 0FB705[50430100]          <1>      movzx   eax, word [DelFile_EntryCounter]
2605 0000AA1C 29D0                    <1>      sub     eax, edx
2606 0000AA1E 72BD                    <1>      jc      short loc_rename_dirent_invd_retn
2607                                     <1>
2608                                     <1> loc_rename_dirent_del_ln_continue:
2609                                     <1>      ; EAX = Directory Entry Number of the long name last entry
2610 0000AA20 E805FDFFFF              <1>      call    delete_longname
2611                                     <1>
2612                                     <1> loc_rename_dirent_update_parent_dir_lm_date:
2613 0000AA25 E83BFCFFFF              <1>      call    update_parent_dir_lmdt
2614 0000AA2A 31C0                    <1>      xor     eax, eax
2615 0000AA2C C3                      <1>      retn
2616                                     <1>
2617                                     <1> move_source_file_to_destination_file:
2618                                     <1>      ; 15/10/2016
2619                                     <1>      ; 11/03/2016
2620                                     <1>      ; 10/03/2016 (TRDOS 386 = TRDOS v2.0)
2621                                     <1>      ; 01/08/2011 (FILE.ASM)
2622                                     <1>      ; 04/08/2010
2623                                     <1>      ;
2624                                     <1>      ; Phase 1 -> Check destination file,
2625                                     <1>      ; 'not found' is required
2626                                     <1>      ; Phase 2 -> Check source file
2627                                     <1>      ; 'found' and proper attributes is required
2628                                     <1>      ; Phase 3 -> Make destination directory entry,

```

```

2629      <1>      ;      add new dir cluster or section if it is required
2630      <1>      ;      Phase 4 -> Delete source directory entry.
2631      <1>      ;      cf = 1 causes to return before the phase 4.
2632      <1>      ;      (source file protection against any possible errors)
2633      <1>      ;
2634      <1>      ; 08/05/2011 major modification
2635      <1>      ;      -> destination file deleting is removed
2636      <1>      ;      for msdos move/rename compatibility.
2637      <1>      ;      (Access denied error will return if
2638      <1>      ;      the destination file is found...)
2639      <1>      ; INPUT ->
2640      <1>      ;      ESI = Source File Pathname (Asciiz)
2641      <1>      ;      EDI = Destination File Pathname (Asciiz)
2642      <1>      ;      AL = 0 --> Interrupt (System call)
2643      <1>      ;      AL > 0 --> Command Interpreter (Question)
2644      <1>      ;      AL = 1 --> Question Phase
2645      <1>      ;      AL = 2 --> Progress Phase
2646      <1>      ; OUTPUT ->
2647      <1>      ;      cf = 0 -> OK
2648      <1>      ;      EAX = Destination directory first cluster
2649      <1>      ;      ESI = Logical DOS drive description table
2650      <1>      ;      EBX = Destination file structure offset
2651      <1>      ;      CX = 0 (CX > 0 --> calculate free space error)
2652      <1>      ;      cf = 1 -> Error code in EAX (AL)
2653      <1>      ;
2654      <1>      ; (EDX, ECX, EBX, ESI, EDI will be changed)
2655      <1>
2656      0000AA2D 3C02      <1>      cmp     al, 2
2657      0000AA2F 0F847F010000      <1>      je      msftdf_df2_check_directory
2658      0000AA35 A2[D2440100]      <1>      mov     [move_cmd_phase], al
2659      <1>
2660      <1> msftdf_parse_sf_path:
2661      <1>      ; ESI = ASCIIIZ pathname (Source)
2662      0000AA3A 57      <1>      push    edi
2663      0000AA3B BF[D0430100]      <1>      mov     edi, SourceFile_Drv
2664      0000AA40 E821F7FFFF      <1>      call   parse_path_name
2665      0000AA45 5E      <1>      pop     esi
2666      0000AA46 7211      <1>      jc      short msftdf_psf_retn
2667      <1>
2668      <1> msftdf_parse_df_path:
2669      <1>      ; ESI = ASCIIIZ pathname (Destination)
2670      0000AA48 BF[50440100]      <1>      mov     edi, DestinationFile_Drv
2671      0000AA4D E814F7FFFF      <1>      call   parse_path_name
2672      0000AA52 7306      <1>      jnc     short msftdf_check_sf_drv
2673      <1>
2674      0000AA54 3C01      <1>      cmp     al, 1 ; File or directory name is not existing
2675      0000AA56 7602      <1>      jna     short msftdf_check_sf_drv
2676      <1>
2677      <1> msftdf_stc_retn:
2678      0000AA58 F9      <1>      stc
2679      <1> msftdf_psf_retn:
2680      0000AA59 C3      <1>      retn
2681      <1>
2682      <1> msftdf_check_sf_drv:
2683      0000AA5A A0[D0430100]      <1>      mov     al, [SourceFile_Drv]
2684      <1>
2685      <1> msftdf_check_df_drv:
2686      0000AA5F 8A15[50440100]      <1>      mov     dl, [DestinationFile_Drv]
2687      <1>
2688      <1> msftdf_compare_sf_df_drv:
2689      0000AA65 29DB      <1>      sub     ebx, ebx
2690      0000AA67 8A3D[E6380100]      <1>      mov     bh, [Current_Drv]
2691      0000AA6D 38C2      <1>      cmp     dl, al
2692      0000AA6F 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      0000AA71 88C6      <1>      mov     dh, al ; destination file's drive number
2697      <1>      ; 15/10/2016 (11h -> 21)
2698      0000AA73 B815000000      <1>      mov     eax, 21 ; Not the same drive
2699      0000AA78 F9      <1>      stc
2700      0000AA79 C3      <1>      retn
2701      <1>
2702      <1> msftdf_check_sf_df_drv_ok:
2703      0000AA7A 8815[D3440100]      <1>      mov     [msftdf_sf_df_drv], dl
2704      <1>
2705      <1>      sub     eax, eax
2706      0000AA82 88D4      <1>      mov     ah, dl
2707      0000AA84 0500010900      <1>      add     eax, Logical_DOSDisks
2708      0000AA89 A3[D4440100]      <1>      mov     [msftdf_drv_offset], eax
2709      <1>
2710      0000AA8E 38FA      <1>      cmp     dl, bh ; byte [Current_Drv]
2711      0000AA90 7407      <1>      je      short msftdf_df_check_directory
2712      <1>
2713      <1> msftdf_change_drv:
2714      0000AA92 E826C1FFFF      <1>      call   change_current_drive
2715      0000AA97 726D      <1>      jc      short msftdf_df_error_retn
2716      <1>
2717      <1> msftdf_check_destination_file:
2718      <1> msftdf_df_check_directory:
2719      0000AA99 BE[51440100]      <1>      mov     esi, DestinationFile_Directory
2720      0000AA9E 803E20      <1>      cmp     byte [esi], 20h
2721      0000AAA1 760F      <1>      jna     short msftdf_df_find_1
2722      <1>
2723      <1> msftdf_df_change_directory:
2724      0000AAA3 FE05[A8FA0000]      <1>      inc     byte [Restore_CDIR]
2725      0000AAA9 30E4      <1>      xor     ah, ah ; CD_COMMAND sign -> 0
2726      0000AAB E8A2F0FFFF      <1>      call   change_current_directory
2727      0000AAB0 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
2732
2733 0000AAB2 BE[92440100]
2734 0000AAB7 803E20
2735 0000AABA 7631
2736
2737
2738 0000AABC 6631C0
2739 0000AABF E88BD4FFFF
2740 0000AAC4 0F838D000000
2741
2742
2743
2744 0000AACA 3C02
2745 0000AACC 7537
2746
2747
2748
2749 0000AACE BE[92440100]
2750 0000AAD3 E837D8FFFF
2751 0000AAD8 7307
2752
2753 0000AADA B81A000000
2754 0000AADF EB24
2755
2756
2757
2758 0000AAE1 BF[A2440100]
2759 0000AAE6 E8B0F5FFFF
2760 0000AAEB EB1A
2761
2762
2763 0000AAED 89F7
2764 0000AAEF 57
2765 0000AAF0 BE[12440100]
2766 0000AAF5 B90C000000
2767
2768 0000Aafa AC
2769 0000Aafb AA
2770 0000AAFC 08C0
2771 0000AAFE 7402
2772 0000AB00 E2F8
2773
2774 0000AB02 5E
2775 0000AB03 EBB7
2776
2777
2778 0000AB05 F9
2779
2780
2781 0000AB06 C3
2782
2783
2784 0000AB07 803D[A8FA0000]00
2785 0000AB0E 760D
2786 0000AB10 8B35[D4440100]
2787 0000AB16 E859C1FFFF
2788 0000AB1B 72E9
2789
2790
2791 0000AB1D BE[D1430100]
2792 0000AB22 803E20
2793 0000AB25 760F
2794
2795 0000AB27 FE05[A8FA0000]
2796 0000AB2D 30E4
2797 0000AB2F E81EF0FFFF
2798 0000AB34 7227
2799
2800
2801
2802
2803
2804 0000AB36 BE[12440100]
2805 0000AB3B 66B80018
2806 0000AB3F E80BD4FFFF
2807 0000AB44 7217
2808
2809
2810 0000AB46 6609D2
2811 0000AB49 7407
2812
2813
2814 0000AB4B B802000000
2815 0000AB50 F9
2816 0000AB51 C3
2817
2818
2819 0000AB52 80E31F
2820 0000AB55 7416
2821
2822
2823 0000AB57 B805000000
2824 0000AB5C F9
2825
2826
2827 0000AB5D C3
2828
2829
2830 0000AB5E 31C0
2831 0000AB60 A2[D2440100]
2832 0000AB65 FEC0

```

```

<1>
<1> msftdf_df_find_1:
<1>     mov     esi, DestinationFile_Name
<1>     cmp     byte [esi], 20h
<1>     jna     short msftdf_df_copy_sf_name
<1>
<1> msftdf_df_find_2:
<1>     xor     ax, ax ; DestinationFile_AttributesMask -> any/zero
<1>     call    find_first_file
<1>     jnc     msftdf_permission_denied_retn
<1>
<1> msftdf_df_check_error_code:
<1>     ;cmp     eax, 2 ; File not found error
<1>     cmp     al, 2
<1>     jne     short msftdf_df_stc_retn
<1>
<1> msftdf_df_check_fname:
<1>     ; 15/10/2016
<1>     mov     esi, DestinationFile_Name ; *
<1>     call    check_filename
<1>     jnc     short msftdf_convert_df_dirententry_name
<1>     ; invalid file name chars !
<1>     mov     eax, ERR_INV_FILE_NAME ; 26
<1>     jmp     short msftdf_df_stc_retn
<1>
<1> msftdf_convert_df_dirententry_name:
<1>     ; mov     esi, DestinationFile_Name ; *
<1>     mov     edi, DestinationFile_DirEntry
<1>     call    convert_file_name
<1>     jmp     short msftdf_restore_current_dir_1
<1>
<1> msftdf_df_copy_sf_name:
<1>     mov     edi, esi
<1>     push    edi
<1>     mov     esi, SourceFile_Name
<1>     mov     ecx, 12
<1> msftdf_df_copy_sf_name_loop:
<1>     lodsb
<1>     stosb
<1>     or      al, al
<1>     jz      short msftdf_df_copy_sf_name_ok
<1>     loop    msftdf_df_copy_sf_name_loop
<1> msftdf_df_copy_sf_name_ok:
<1>     pop     esi
<1>     jmp     short msftdf_df_find_2
<1>
<1> msftdf_df_stc_retn:
<1>     stc
<1> msftdf_restore_cdir_failed:
<1> msftdf_df_error_retn:
<1>     retn
<1>
<1> msftdf_restore_current_dir_1:
<1>     cmp     byte [Restore_CDIRE], 0
<1>     jna     short msftdf_sf_check_directory
<1>     mov     esi, [msftdf_drv_offset]
<1>     call    restore_current_directory
<1>     jc      short msftdf_restore_cdir_failed
<1>
<1> msftdf_sf_check_directory:
<1>     mov     esi, SourceFile_Directory
<1>     cmp     byte [esi], 20h
<1>     jna     short msftdf_sf_find
<1> msftdf_sf_change_directory:
<1>     inc     byte [Restore_CDIRE]
<1>     xor     ah, ah ; CD_COMMAND sign -> 0
<1>     call    change_current_directory
<1>     jc      short msftdf_return
<1>
<1> ;msftdf_sf_change_prompt_dir_string:
<1> ;     call    change_prompt_dir_string
<1>
<1> msftdf_sf_find:
<1>     mov     esi, SourceFile_Name ; Offset 66
<1>     mov     ax, 1800h ; Only files
<1>     call    find_first_file
<1>     jc      short msftdf_return
<1>
<1> msftdf_sf_ambgfn_check:
<1>     or      dx, dx ; Ambiguous filename chars used sign (DX>0)
<1>     jz      short msftdf_sf_found
<1>
<1> msftdf_ambiguous_file_name_error:
<1>     mov     eax, 2 ; File not found error
<1>     stc
<1>     retn
<1>
<1> msftdf_sf_found:
<1>     and     bl, 1Fh ; Attributes, D-V-S-H-R
<1>     jz      short msftdf_save_sf_structure
<1>
<1> msftdf_permission_denied_retn:
<1>     mov     eax, 05h ; Access (Permission) denied !
<1>     stc
<1> msftdf_rest_cdir_err_retn:
<1> msftdf_return:
<1>     retn
<1>
<1> msftdf_phase_1_return:
<1>     xor     eax, eax
<1>     mov     [move_cmd_phase], al ; 0
<1>     inc     al ; mov al, 1

```

```

2833 0000AB67 BB[B4AB0000]    <1>      mov     ebx, msftdf_df2_check_directory
2834                                <1>      ;mov     edx, 0FFFFFFFh
2835 0000AB6C C3              <1>      retn
2836                                <1>
2837                                <1> msftdf_save_sf_structure:
2838 0000AB6D BE[DC420100]    <1>      mov     esi, FindFile_DirEntry
2839 0000AB72 BF[22440100]    <1>      mov     edi, SourceFile_DirEntry
2840 0000AB77 B908000000      <1>      mov     ecx, 8
2841 0000AB7C F3A5            <1>      rep     movsd
2842                                <1>
2843                                <1> msftdf_df_copy_sf_parameters:
2844 0000AB7E BE0B000000      <1>      mov     esi, 11
2845 0000AB83 89F7            <1>      mov     edi, esi
2846 0000AB85 81C6[22440100] <1>      add     esi, SourceFile_DirEntry
2847 0000AB8B 81C7[A2440100] <1>      add     edi, DestinationFile_DirEntry
2848                                <1>      ;mov     ecx, 21
2849 0000AB91 B115            <1>      mov     cl, 21
2850 0000AB93 F3A4            <1>      rep     movsb
2851                                <1>
2852                                <1> msftdf_restore_current_dir_2:
2853 0000AB95 803D[A8FA0000]00 <1>      cmp     byte [Restore_CDIR], 0
2854 0000AB9C 760D            <1>      jna     short msftdf_df2_check_move_cmd_phase
2855 0000AB9E 8B35[D4440100] <1>      mov     esi, [msftdf_drv_offset]
2856 0000ABA4 E8CBC0FFFF      <1>      call    restore_current_directory
2857 0000ABA9 72B2            <1>      jc      short msftdf_rest_cdir_err_retn
2858                                <1>
2859                                <1> msftdf_df2_check_move_cmd_phase:
2860 0000ABAB 803D[D2440100]01 <1>      cmp     byte [move_cmd_phase], 1
2861 0000ABB2 74AA            <1>      je      short msftdf_phase_1_return
2862                                <1>
2863                                <1> msftdf_df2_check_directory:
2864 0000ABB4 BE[51440100]    <1>      mov     esi, DestinationFile_Directory
2865 0000ABB9 803E20          <1>      cmp     byte [esi], 20h
2866 0000ABBC 760F            <1>      jna     short msftdf_make_dfde_locate_ffc_on_directory
2867                                <1> msftdf_df2_change_directory:
2868 0000ABBE FE05[A8FA0000] <1>      inc     byte [Restore_CDIR]
2869 0000ABC4 30E4            <1>      xor     ah, ah ; CD_COMMAND sign -> 0
2870 0000ABC6 E887EFFFFF      <1>      call    change_current_directory
2871 0000ABCB 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_ffc_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 0000ABCD 31C0            <1>      xor     eax, eax
2883 0000ABCF 89C1            <1>      mov     ecx, eax
2884 0000ABD1 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 0000ABD3 E8CEF1FFFF      <1>      call    locate_current_dir_file
2889 0000ABD8 733F            <1>      jnc     msftdf_make_dfde_set_ff_dir_entry
2890                                <1>
2891                                <1>      ;cmp     eax, 2
2892 0000ABDA 3C02            <1>      cmp     al, 2
2893 0000ABDC 7537            <1>      jne     short msftdf_error_retn
2894                                <1>
2895                                <1> msftdf_add_new_dir_entry_check_fs:
2896 0000ABDE 8B35[D4440100] <1>      mov     esi, [msftdf_drv_offset]
2897 0000ABE4 A1[15410100]    <1>      mov     eax, [DirBuff_Cluster]
2898 0000ABE9 807E0300      <1>      cmp     byte [esi+LD_FATType], 0
2899 0000ABED 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 0000ABEF 30ED            <1>      xor     ch, ch ; cx = 0 --> add a new subdir section
2905 0000ABF1 E8830C0000      <1>      call    add_new_fs_section
2906 0000ABF6 721E            <1>      jc      short msftdf_dsfd_error_retn
2907                                <1>      ;mov     [createfile_LastDirCluster], eax
2908                                <1>
2909 0000ABF8 E8A30E0000      <1>      call    load_FS_sub_directory
2910                                <1>      ;mov     ebx, Directory_Buffer
2911 0000ABFD 7318            <1>      jnc     short msftdf_add_new_fs_subdir_section_ok
2912 0000ABFF C3              <1>      retn
2913                                <1>
2914                                <1> msftdf_add_new_subdir_cluster:
2915 0000AC00 E881150000      <1>      call    add_new_cluster
2916 0000AC05 720F            <1>      jc      short msftdf_dsfd_error_retn
2917                                <1>
2918                                <1>      ;mov     [createfile_LastDirCluster], eax
2919                                <1>
2920 0000AC07 E8570E0000      <1>      call    load_FAT_sub_directory
2921 0000AC0C 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 0000AC0E 50              <1>      push    eax
2927 0000AC0F E851FAFFFF      <1>      call    update_parent_dir_lmdt
2928 0000AC14 58              <1>      pop     eax
2929                                <1>
2930                                <1> msftdf_error_retn:
2931 0000AC15 F9              <1>      stc
2932                                <1> msftdf_dsfd_restore_cdir_failed:
2933                                <1> msftdf_dsfd_error_retn:
2934 0000AC16 C3              <1>      retn

```



```

2935 <1>
2936 <1> msftdf_add_new_fs_subdir_section_ok:
2937 <1> msftdf_add_new_subdir_cluster_ok:
2938 0000AC17 89DF <1> mov edi, ebx ; Directory buffer address
2939 <1>
2940 <1> msftdf_make_dfde_set_ff_dir_entry:
2941 0000AC19 8B15[E0380100] <1> mov edx, [Current_Dir_FCluster]
2942 0000AC1F 8915[38450100] <1> mov [createfile_FFCluster], edx
2943 <1> ; EDI = Directory entry offset
2944 0000AC25 BE[A2440100] <1> mov esi, DestinationFile_DirEntry
2945 0000AC2A B908000000 <1> mov ecx, 8
2946 0000AC2F F3A5 <1> rep movsd
2947 <1>
2948 0000AC31 C605[10410100]02 <1> mov byte [DirBuff_ValidData], 2
2949 0000AC38 E88DF9FFFF <1> call save_directory_buffer
2950 0000AC3D 72CF <1> jc short msftdf_make_dfde_err_upd_pdir_lmdt
2951 <1>
2952 <1> msftdf_make_dfde_update_pdir_lmdt:
2953 0000AC3F E821FAFFFF <1> call update_parent_dir_lmdt
2954 <1>
2955 <1> msftdf_dsfd restore_current_dir_1:
2956 0000AC44 803D[A8FA0000]00 <1> cmp byte [Restore_CDIR], 0
2957 0000AC4B 760D <1> jna short msftdf_dsfd_check_directory
2958 0000AC4D 8B35[D4440100] <1> mov esi, [msftdf_drv_offset]
2959 0000AC53 E81CC0FFFF <1> call restore_current_directory
2960 0000AC58 72BC <1> jc short msftdf_dsfd_restore_cdir_failed
2961 <1>
2962 <1> msftdf_dsfd_check_directory:
2963 0000AC5A BE[D1430100] <1> mov esi, SourceFile_Directory
2964 0000AC5F 803E20 <1> cmp byte [esi], 20h
2965 0000AC62 760F <1> jna short msftdf_dsfd_find_file
2966 <1>
2967 <1> msftdf_dsfd_change_directory:
2968 0000AC64 FE05[A8FA0000] <1> inc byte [Restore_CDIR]
2969 0000AC6A 28E4 <1> sub ah, ah ; CD_COMMAND sign -> 0
2970 0000AC6C E8E1EEFFFF <1> call change_current_directory
2971 0000AC71 72A3 <1> jc short msftdf_dsfd_error_retn
2972 <1>
2973 <1> ;msftdf_dsfd_sf_change_prompt_dir_string:
2974 <1> ; call change_prompt_dir_string
2975 <1>
2976 <1> msftdf_dsfd_find_file:
2977 0000AC73 BE[12440100] <1> mov esi, SourceFile_Name ; Offset 66
2978 0000AC78 668B460E <1> mov ax, [esi+14] ; 80 -> SourceFile_AttributesMask
2979 0000AC7C E8CED2FFFF <1> call find_first_file
2980 0000AC81 7293 <1> jc short msftdf_dsfd_error_retn
2981 <1>
2982 <1> msftdf_dsfd_delete_direntry:
2983 0000AC83 8B35[D4440100] <1> mov esi, [msftdf_drv_offset]
2984 <1>
2985 0000AC89 807E0300 <1> cmp byte [esi+LD_FATType], 0
2986 0000AC8D 770A <1> ja short msftdf_delete_FAT_direntry
2987 <1>
2988 0000AC8F 30DB <1> xor bl, bl
2989 <1> ; BL = 0 -> File
2990 <1> ; EDI -> Directory buffer entry offset/address
2991 0000AC91 E8E40B0000 <1> call delete_fs_directory_entry
2992 0000AC96 7315 <1> jnc short msftdf_dsfd_restore_current_dir_2
2993 0000AC98 C3 <1> retn
2994 <1>
2995 <1> msftdf_delete_FAT_direntry:
2996 0000AC99 8A1D[D9420100] <1> mov bl, [FindFile_LongNameEntryLength]
2997 0000AC9F 668B0D[04430100] <1> mov cx, [FindFile_DirEntryNumber]
2998 <1> ; ESI = Logical DOS drive description table address
2999 <1> ; EDI = Directory buffer entry offset/address
3000 0000ACA6 E899FCFFFF <1> call delete_directory_entry
3001 0000ACAB 721C <1> jc short msftdf_retn
3002 <1>
3003 <1> msftdf_dsfd_restore_current_dir_2:
3004 0000ACAD 803D[A8FA0000]00 <1> cmp byte [Restore_CDIR], 0
3005 0000ACB4 7607 <1> jna short msftdf_new_dir_fcluster_retn
3006 <1> ;mov esi, [msftdf_drv_offset]
3007 0000ACB6 E8B9BFFFFF <1> call restore_current_directory
3008 0000ACBB 720C <1> jc short msftdf_retn
3009 <1>
3010 <1> msftdf_new_dir_fcluster_retn:
3011 0000ACBD 31C9 <1> xor ecx, ecx
3012 0000ACBF A1[38450100] <1> mov eax, [createfile_FFCluster]
3013 0000ACC4 BB[50440100] <1> mov ebx, DestinationFile_Drv
3014 <1>
3015 <1> msftdf_retn:
3016 0000ACC9 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      0000ACCA 3C02      <1>      cmp     al, 2
3079      0000ACCC 0F845A020000 <1>      je      csftdf2_check_cdrv
3080      <1>
3081      <1> ; Phase 1
3082      <1>
3083      0000ACD2 A2[F8440100] <1>      mov     byte [copy_cmd_phase], al
3084      <1>
3085      0000ACD7 57      <1>      push    edi ; *
3086      <1>
3087      <1> csftdf_parse_sf_path:
3088      0000ACD8 BF[D0430100] <1>      mov     edi, SourceFile_Drv
3089      0000ACDD E884F4FFFF <1>      call    parse_path_name
3090      0000ACE2 721C      <1>      jc      short csftdf_parse_sf_path_failed
3091      <1>
3092      <1> csftdf_parse_df_path:
3093      0000ACE4 5E      <1>      pop     esi ; * (pushed edi)
3094      <1>
3095      <1> csftdf_sf_check_filename_exists:
3096      0000ACE5 803D[12440100]21 <1>      cmp     byte [SourceFile_Name], 21h
3097      0000ACEC 7215      <1>      jb      short csftdf_sf_file_not_found_error
3098      <1>
3099      0000ACEE BF[50440100] <1>      mov     edi, DestinationFile_Drv
3100      0000ACF3 E86EF4FFFF <1>      call    parse_path_name
3101      0000ACF8 7310      <1>      jnc     short csftdf_check_sf_cdrv
3102      <1>
3103      0000ACFA 3C01      <1>      cmp     al, 1 ; File or directory name is not existing
3104      0000ACFC 760C      <1>      jna     short csftdf_check_sf_cdrv
3105      <1>
3106      <1> csftdf_parse_df_path_failed:
3107      0000ACFE F9      <1>      stc
3108      <1> csftdf_sf_error_retn:
3109      0000ACFF C3      <1>      retn
3110      <1>
3111      <1> csftdf_parse_sf_path_failed:
3112      0000AD00 5F      <1>      pop     edi ; *
3113      0000AD01 EBFC      <1>      jmp     short csftdf_sf_error_retn
3114      <1>
3115      <1> csftdf_sf_file_not_found_error:
3116      0000AD03 B802000000 <1>      mov     eax, 2 ; File not found
3117      0000AD08 EBF5      <1>      jmp     short csftdf_sf_error_retn
3118      <1>
3119      <1> csftdf_check_sf_cdrv:
3120      0000AD0A 8A3D[E6380100] <1>      mov     bh, [Current_Drv]
3121      <1>
3122      0000AD10 883D[FB440100] <1>      mov     [csftdf_cdrv], bh ; 23/03/2016
3123      <1>
3124      0000AD16 8A15[D0430100] <1>      mov     dl, [SourceFile_Drv]
3125      0000AD1C 38FA      <1>      cmp     dl, bh ; byte [Current_Drv]
3126      0000AD1E 7407      <1>      je      short csftdf_sf_check_directory
3127      <1>
3128      0000AD20 E898BEFFFF <1>      call    change_current_drive
3129      0000AD25 72D8      <1>      jc      short csftdf_sf_error_retn
3130      <1>
3131      <1> csftdf_sf_check_directory:
3132      0000AD27 BE[D1430100] <1>      mov     esi, SourceFile_Directory
3133      0000AD2C 803E20 <1>      cmp     byte [esi], 20h
3134      0000AD2F 760F <1>      jna     short csftdf_find_sf
3135      <1>
3136      <1> csftdf_sf_change_directory:
3137      0000AD31 FE05[A8FA0000] <1>      inc     byte [Restore_CDIRE]
3138      0000AD37 30E4 <1>      xor     ah, ah ; CD_COMMAND sign -> 0

```

```

3139 0000AD39 E814EEFFFF <1> call change_current_directory
3140 0000AD3E 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 0000AD40 BE[12440100] <1> mov esi, SourceFile_Name
3147 0000AD45 66B80018 <1> mov ax, 1800h ; Except volume label and dirs
3148 0000AD49 E801D2FFFF <1> call find_first_file
3149 0000AD4E 72AF <1> jc short csftdf_sf_error_retn
3150 <1>
3151 <1> csftdf_sf_ambgfn_check:
3152 0000AD50 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)
3153 0000AD53 7407 <1> jz short csftdf_sf_found
3154 <1>
3155 <1> csftdf_ambiguous_file_name_error:
3156 0000AD55 B802000000 <1> mov eax, 2 ; File not found error
3157 0000AD5A F9 <1> stc
3158 0000AD5B C3 <1> retn
3159 <1>
3160 <1> csftdf_sf_found:
3161 0000AD5C A3[FC440100] <1> mov [csftdf_filesized], eax
3162 <1>
3163 0000AD61 09C0 <1> or eax, eax
3164 0000AD63 7507 <1> jnz short csftdf_set_source_file_dirent
3165 <1>
3166 <1> csftdf_sf_file_size_zero:
3167 0000AD65 B814000000 <1> mov eax, 20 ; TRDOS zero length (file size) error
3168 0000AD6A F9 <1> stc
3169 0000AD6B C3 <1> retn
3170 <1>
3171 <1> csftdf_set_source_file_dirent:
3172 0000AD6C BE[DC420100] <1> mov esi, FindFile_DirEntry
3173 0000AD71 BF[22440100] <1> mov edi, SourceFile_DirEntry
3174 0000AD76 B908000000 <1> mov ecx, 8
3175 0000AD7B F3A5 <1> rep movsd
3176 <1>
3177 <1> csftdf_sf_restore_cdrv:
3178 <1> ; 22/03/2016
3179 0000AD7D 8A15[FB440100] <1> mov dl, [csftdf_cdrv]
3180 0000AD83 3A15[E6380100] <1> cmp dl, [Current_Drv]
3181 0000AD89 7407 <1> je short csftdf_sf_restore_cdir
3182 0000AD8B E82DBEFFFF <1> call change_current_drive
3183 0000AD90 724F <1> jc short csftdf_df_error_retn ; 30/03/2016
3184 <1>
3185 <1> csftdf_sf_restore_cdir:
3186 0000AD92 803D[A8FA0000]00 <1> cmp byte [Restore_CDIRE], 0
3187 0000AD99 7612 <1> jna short csftdf_df_check_filename_exists
3188 0000AD9B 29C0 <1> sub eax, eax
3189 0000AD9D BE00010900 <1> mov esi, Logical_DOSDisks
3190 0000ADA2 88D4 <1> mov ah, dl ; byte [csftdf_cdrv]
3191 0000ADA4 01C6 <1> add esi, eax
3192 0000ADA6 E8C9BEFFFF <1> call restore_current_directory
3193 0000ADAB 7234 <1> jc short csftdf_df_error_retn
3194 <1>
3195 <1> csftdf_df_check_filename_exists:
3196 0000ADAD 803D[92440100]20 <1> cmp byte [DestinationFile_Name], 20h
3197 0000ADB4 7716 <1> ja short csftdf_check_df_cdrv
3198 <1>
3199 <1> csftdf_copy_sf_name:
3200 0000ADB6 BF[92440100] <1> mov edi, DestinationFile_Name
3201 0000ADBB BE[12440100] <1> mov esi, SourceFile_Name
3202 0000ADC0 B10C <1> mov cl, 12
3203 <1>
3204 <1> csftdf_df_copy_sf_name_loop:
3205 0000ADC2 AC <1> lodsb
3206 0000ADC3 AA <1> stosb
3207 0000ADC4 08C0 <1> or al, al
3208 0000ADC6 7404 <1> jz short csftdf_check_df_cdrv
3209 0000ADC8 FEC9 <1> dec cl
3210 0000ADCA 75F6 <1> jnz csftdf_df_copy_sf_name_loop
3211 <1>
3212 <1> csftdf_check_df_cdrv:
3213 0000ADCC 8A15[50440100] <1> mov dl, [DestinationFile_Drv]
3214 0000ADD2 3A15[E6380100] <1> cmp dl, [Current_Drv]
3215 0000ADD8 7408 <1> je short csftdf_df_check_directory
3216 <1>
3217 0000ADDA E8DEBDFFFF <1> call change_current_drive
3218 0000ADDF 7301 <1> jnc short csftdf_df_check_directory
3219 <1>
3220 <1> csftdf_df_error_retn:
3221 0000ADE1 C3 <1> retn
3222 <1>
3223 <1> csftdf_df_check_directory:
3224 0000ADE2 BE[51440100] <1> mov esi, DestinationFile_Directory
3225 0000ADE7 803E20 <1> cmp byte [esi], 20h
3226 0000ADEA 760F <1> jna short csftdf_find_df
3227 <1>
3228 <1> csftdf_df_change_directory:
3229 0000ADEC FE05[A8FA0000] <1> inc byte [Restore_CDIRE]
3230 0000ADF2 28E4 <1> sub ah, ah ; CD_COMMAND sign -> 0
3231 0000ADF4 E859EDFFFF <1> call change_current_directory
3232 0000ADF9 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 0000ADFB 29DB <1> sub ebx, ebx
3240 0000ADFD 8A3D[50440100] <1> mov bh, [DestinationFile_Drv]

```

```

3241 0000AE03 81C300010900 <1> add ebx, Logical_DOSDisks
3242 0000AE09 891D[28450100] <1> mov [csftdf_df_drv_dt], ebx
3243 <1>
3244 0000AE0F BE[92440100] <1> mov esi, DestinationFile_Name
3245 0000AE14 6631C0 <1> xor ax, ax
3246 <1> ; DestinationFile_AttributesMask -> any/zero
3247 0000AE17 E833D1FFFF <1> call find_first_file
3248 0000AE1C 7218 <1> jc short csftdf_df_check_error_code
3249 <1>
3250 <1> csftdf_df_ambgfn_check:
3251 0000AE1E 6609D2 <1> or dx, dx ; Ambiguous filename chars used sign (DX>0)
3252 0000AE21 752A <1> jnz short csftdf_df_error_inv_fname
3253 <1>
3254 <1> csftdf_df_found:
3255 0000AE23 C605[FA440100]01 <1> mov byte [DestinationFileFound], 1
3256 <1> ; 17/10/2016 (cl -> bl)
3257 0000AE2A 80E31F <1> and bl, 1Fh ; Attributes, D-V-S-H-R
3258 0000AE2D 745F <1> jz short csftdf_df_save_first_cluster
3259 <1>
3260 <1> csftdf_df_permission_denied_retn:
3261 0000AE2F B805000000 <1> mov eax, 05h ; Access/Permisson denied.
3262 <1> csftdf_df_error_stc_retn:
3263 0000AE34 F9 <1> stc
3264 0000AE35 C3 <1> retn
3265 <1>
3266 <1> csftdf_df_check_error_code:
3267 <1> ;cmp eax, 2
3268 0000AE36 3C02 <1> cmp al, 2
3269 0000AE38 75FA <1> jne short csftdf_df_error_stc_retn
3270 <1>
3271 0000AE3A C605[FA440100]00 <1> mov byte [DestinationFileFound], 0
3272 <1>
3273 <1> ; 15/10/2016
3274 0000AE41 BE[CC420100] <1> mov esi, FindFile_Name ; *
3275 0000AE46 E8C4D4FFFF <1> call check_filename
3276 0000AE4B 7307 <1> jnc short csftdf_df_valid_fname
3277 <1> csftdf_df_error_inv_fname: ; 'invalid file name !'
3278 0000AE4D B81A000000 <1> mov eax, ERR_INV_FILE_NAME ; 26
3279 0000AE52 F9 <1> stc
3280 0000AE53 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 0000AE54 BF[92440100] <1> mov edi, DestinationFile_Name
3287 0000AE59 A5 <1> movsd
3288 0000AE5A A5 <1> movsd
3289 0000AE5B A5 <1> movsd
3290 <1> ;movsb
3291 <1>
3292 <1> csftdf_check_disk_free_size_0:
3293 0000AE5C A1[3E440100] <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 0000AE61 8B35[28450100] <1> mov esi, [csftdf_df_drv_dt] ; 23/03/2016
3302 <1>
3303 0000AE67 0FB74E11 <1> movzx ecx, word [esi+LD_BPB+BytesPerSec] ; 17, LD_BPB + 0Bh
3304 0000AE6B 01C8 <1> add eax, ecx
3305 0000AE6D 48 <1> dec eax ; file size (additional bytes) + 511 (round up)
3306 <1> csftdf_check_disk_free_size_3: ; 16/03/2016
3307 0000AE6E 29D2 <1> sub edx, edx
3308 0000AE70 F7F1 <1> div ecx ; bytes per sector
3309 <1>
3310 <1> csftdf_check_disk_free_size:
3311 0000AE72 3B4674 <1> cmp eax, [esi+LD_FreeSectors]
3312 0000AE75 0F8294000000 <1> jb csftdf_check_disk_free_size_ok
3313 0000AE7B 770A <1> ja short csftdf_df_insufficient_disk_space
3314 <1>
3315 0000AE7D 807E0300 <1> cmp byte [esi+LD_FATType], 0 ; FS needs FDT sector also.
3316 0000AE81 0F8788000000 <1> ja csftdf_check_disk_free_size_ok
3317 <1>
3318 <1> csftdf_df_insufficient_disk_space:
3319 0000AE87 B827000000 <1> mov eax, 27h ; insufficient disk space
3320 0000AE8C 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 0000AE8E 81EF00000800 <1> sub edi, Directory_Buffer ; (<65536)
3328 0000AE94 66C1EF05 <1> shr di, 5 ; Convert entry offset to entry index/number
3329 0000AE98 66893D[CA440100] <1> mov [DestinationFile_DirEntryNumber], di ; (<2048)
3330 <1>
3331 <1> csftdf_df_check_sf_df_fcluster:
3332 0000AE9F 668B5614 <1> mov dx, [esi+DirEntry_FstClusHI]
3333 0000AEA3 C1E210 <1> shl edx, 16
3334 0000AEA6 668B561A <1> mov dx, [esi+DirEntry_FstClusLO]
3335 0000AEAA 8915[0C450100] <1> mov [csftdf_df_cluster], edx
3336 <1> csftdf_df_check_sf_df_fcluster_1:
3337 0000AEB0 668B15[36440100] <1> mov dx, [SourceFile_DirEntry+DirEntry_FstClusHI]
3338 0000AEB7 C1E210 <1> shl edx, 16
3339 0000AEB8 668B15[3C440100] <1> mov dx, [SourceFile_DirEntry+DirEntry_FstClusLO]
3340 0000AEC1 3B15[0C450100] <1> cmp edx, [csftdf_df_cluster]
3341 0000AEC7 7512 <1> jne short csftdf_df_check_sf_df_fcluster_ok
3342 <1> csftdf_df_check_sf_df_drv:

```



```

3343 0000AEC9 8A15[D0430100] <1> mov dl, [SourceFile_Drv]
3344 0000AECF 3A15[50440100] <1> cmp dl, [DestinationFile_Drv]
3345 0000AED5 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 0000AED7 31C0 <1> xor eax, eax ; mov eax, 0 -> Bad command or file name !
3351 0000AED9 F9 <1> stc
3352 0000AEDA 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 0000AEDB BF[A2440100] <1> mov edi, DestinationFile_DirEntry
3358 0000AEE0 B908000000 <1> mov ecx, 8
3359 0000AEE5 F3A5 <1> rep movsd
3360 <1>
3361 <1> csftdf_check_disk_free_size_2:
3362 0000AEE7 89C2 <1> mov edx, eax ; Old destination file size
3363 <1>
3364 <1> ;mov eax, [SourceFile_DirEntry+DirEntry_FileSize]
3365 0000AEE9 A1[FC440100] <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 0000AEEE 8B35[28450100] <1> mov esi, [csftdf_df_drv_dt] ; 23/03/2016
3375 <1>
3376 0000AEF4 668B4E11 <1> mov cx, [esi+LD_BPB+BytesPerSec] ; 17, LD_BPB + 0Bh
3377 0000AEF8 01CA <1> add edx, ecx ; + 512
3378 0000AEFA 01C8 <1> add eax, ecx ; + 512
3379 0000AEFC 4A <1> dec edx ; old file size + 511 (round up)
3380 0000AEFD 48 <1> dec eax ; new file size + 511 (round up)
3381 0000AEFE F7D9 <1> neg ecx ; -512 ; 0FFFFFFE00h
3382 0000AF00 21CA <1> and edx, ecx ; = old sector count * 512
3383 0000AF02 21C8 <1> and eax, ecx ; = new sector count * 512
3384 <1>
3385 0000AF04 29D0 <1> sub eax, edx ; new file size - old file size (on disk)
3386 0000AF06 7607 <1> jna short csftdf_check_disk_free_size_ok
3387 <1>
3388 0000AF08 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
3391 <1> ; esi = Logical DOS drive number (of destination disk)
3392 0000AF0A E95FFFFFFF <1> jmp csftdf_check_disk_free_size_3
3393 <1>
3394 <1> csftdf_check_disk_free_size_ok:
3395 <1> ; 18/03/2016
3396 <1> csftdf_df_check_copy_cmd_phase:
3397 0000AF0F A0[F8440100] <1> mov al, [copy_cmd_phase]
3398 0000AF14 3C01 <1> cmp al, 1
3399 0000AF16 7514 <1> jne short csftdf2_check_cdrv
3400 <1>
3401 0000AF18 31C0 <1> xor eax, eax
3402 0000AF1A A2[F8440100] <1> mov [copy_cmd_phase], al ; 0
3403 <1>
3404 0000AF1F 8A15[FA440100] <1> mov dl, [DestinationFileFound]
3405 0000AF25 8A35[2D440100] <1> mov dh, [SourceFile_DirEntry+11] ; Attributes
3406 <1>
3407 <1> csftdf_return:
3408 0000AF2B C3 <1> retn
3409 <1>
3410 <1> ; Phase 2
3411 <1>
3412 <1> csftdf2_check_cdrv:
3413 <1> ; 18/03/2016
3414 <1> ; Here, destination drive and directory are ready !
3415 <1> ; (checking/restoring is not needed)
3416 <1> ; (Since at the end of the phase 1)
3417 <1>
3418 <1> ; mov dl, [DestinationFile_Drv]
3419 <1> ; cmp dl, [Current_Drv]
3420 <1> ; je short csftdf2_df_check_directory
3421 <1> ;
3422 <1> ; call change_current_drive
3423 <1> ; jc short csftdf2_read_error
3424 <1> ;
3425 <1> ;csftdf2_df_check_directory:
3426 <1> ; mov esi, DestinationFile_Directory
3427 <1> ; cmp byte [esi], 20h
3428 <1> ; jna short csftdf2_df_check_found_or_not
3429 <1> ;
3430 <1> ;csftdf2_df_change_directory:
3431 <1> ; inc byte [Restore_CDIRE]
3432 <1> ; xor ah, ah ; CD_COMMAND sign -> 0
3433 <1> ; call change_current_directory
3434 <1> ; jc short csftdf2_stc_return
3435 <1> ;
3436 <1> ;csftdf2_df_change_prompt_dir_string:
3437 <1> ; call change_prompt_dir_string
3438 <1> ;
3439 <1> csftdf2_df_check_found_or_not:
3440 <1> ; 21/03/2016
3441 0000AF2C 803D[FA440100]00 <1> cmp byte [DestinationFileFound], 0
3442 0000AF33 7739 <1> ja short csftdf2_set_sf_percentage
3443 <1>
3444 <1> csftdf2_create_file:

```

```

3445 0000AF35 BE[92440100]      <1>      mov     esi, DestinationFile_Name
3446 0000AF3A A1[FC440100]      <1>      mov     eax, [csftdf_filesize]
3447 0000AF3F 30C9             <1>      xor     cl, cl ; 0
3448                               <1>
3449 0000AF41 31DB             <1>      xor     ebx, ebx ; 0
3450 0000AF43 4B              <1>      dec     ebx ; 0FFFFFFFh
3451                               <1>
3452                               <1>      ; INPUT ->
3453                               <1>      ;     EAX -> File Size
3454                               <1>      ;     ESI = ASCIIZ File name
3455                               <1>      ;     CL = File attributes
3456                               <1>      ;     EBX = FFFFFFFFh -> empty file sign for FAT fs
3457                               <1>      ;     EBX <> FFFFFFFFh -> use file size for FAT fs
3458                               <1>      ;
3459                               <1>      ; OUTPUT ->
3460                               <1>      ;     EAX = New file's first cluster
3461                               <1>      ;     ESI = Logical Dos Drv Descr. Table Addr.
3462                               <1>      ;     EBX = CreateFile_Size address
3463                               <1>      ;     ECX = Sectors per cluster (<256)
3464                               <1>      ;     EDX = Directory Entry Index/Number (<65536)
3465                               <1>      ;
3466                               <1>      ;     cf = 1 -> error code in AL (EAX)
3467                               <1>
3468 0000AF44 E8EC050000        <1>      call    create_file
3469                               <1>      ;pop     esi
3470 0000AF49 0F82A3050000        <1>      jc      csftdf2_rw_error
3471                               <1>
3472                               <1> csftdf2_create_file_OK:
3473 0000AF4F A3[0C450100]      <1>      mov     [csftdf_df_cluster], eax
3474                               <1>
3475                               <1>      ; 24/03/2016
3476 0000AF54 668915[CA440100] <1>      mov     [DestinationFile_DirEntryNumber], dx
3477                               <1>
3478                               <1>      ; 21/03/2016
3479 0000AF5B BE00000800        <1>      mov     esi, Directory_Buffer
3480 0000AF60 C1E205             <1>      shl     edx, 5 ; 32 * index number
3481 0000AF63 01D6             <1>      add     esi, edx
3482 0000AF65 BF[A2440100]      <1>      mov     edi, DestinationFile_DirEntry
3483 0000AF6A B108             <1>      mov     cl, 8 ; 32 bytes
3484 0000AF6C F3A5             <1>      rep     movsd
3485                               <1>
3486                               <1> csftdf2_set_sf_percentage:
3487                               <1>      ; 17/03/2016
3488 0000AF6E 31C0             <1>      xor     eax, eax
3489 0000AF70 A2[20450100]      <1>      mov     [csftdf_percentage], al ; 0, reset
3490                               <1>
3491 0000AF75 A3[18450100]      <1>      mov     [csftdf_sf_rbytes], eax ; 0, reset
3492 0000AF7A A3[1C450100]      <1>      mov     [csftdf_df_wbytes], eax ; 0, reset
3493                               <1>
3494 0000AF7F 8A25[D0430100]      <1>      mov     ah, [SourceFile_Drv]
3495 0000AF85 BE00010900        <1>      mov     esi, Logical_DOSDisks
3496 0000AF8A 01C6             <1>      add     esi, eax
3497                               <1>
3498 0000AF8C 8935[24450100]      <1>      mov     [csftdf_sf_drv_dt], esi ; 23/03/2016
3499                               <1>
3500 0000AF92 668B15[36440100]      <1>      mov     dx, [SourceFile_DirEntry+DirEntry_FstClusHI]
3501 0000AF99 C1E210             <1>      shl     edx, 16
3502 0000AF9C 668B15[3C440100]      <1>      mov     dx, [SourceFile_DirEntry+DirEntry_FstClusLO]
3503 0000AFA3 8915[08450100]      <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 0000AFA9 0FB64E13          <1>      movzx   ecx, byte [esi+LD_BPB+SecPerClust]
3512 0000AFAD 880D[4E440100]      <1>      mov     [SourceFile_SecPerClust], cl
3513                               <1>
3514                               <1>      ; 17/03/2016
3515 0000AFB3 38E03             <1>      cmp     [esi+LD_FATType], ch ; 0
3516 0000AFB6 7707             <1>      ja      short csftdf2_set_sf_percent_rsize1
3517                               <1>
3518 0000AFB8 B800000100        <1>      mov     eax, 65536 ; read/write buffer size for Singlix FS
3519 0000AFBD EB06             <1>      jmp     short csftdf2_set_sf_percent_rsize2
3520                               <1>
3521                               <1> csftdf2_set_sf_percent_rsize1:
3522 0000AFBF 668B4611          <1>      mov     ax, [esi+LD_BPB+BytesPerSec]
3523 0000AFC3 F7E1             <1>      mul     ecx
3524                               <1>      ;sub     edx, edx
3525                               <1> csftdf2_set_sf_percent_rsize2:
3526 0000AFC5 A3[10450100]      <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 0000AFCA 8B3D[28450100]      <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 0000AFD0 8A4F13          <1>      mov     cl, [edi+LD_BPB+SecPerClust]
3545 0000AFD3 880D[CE440100]      <1>      mov     [DestinationFile_SecPerClust], cl
3546                               <1>

```

```

3547      <1>      ; 17/03/2016
3548      <1>      cmp     [edi+LD_FATType], ch ; 0
3549      <1>      ja      short csftdf2_set_df_percent_wsize1
3550      <1>
3551      <1>      mov     eax, 65536 ; read/write buffer size for Singlix FS
3552      <1>      jmp     short csftdf2_set_df_percent_wsize2
3553      <1>
3554      <1>      csftdf2_set_df_percent_wsize1:
3555      <1>      movzx   eax, word [edi+LD_BPB+BytesPerSec]
3556      <1>      mul     ecx
3557      <1>      ;sub     edx, edx
3558      <1>      csftdf2_set_df_percent_wsize2:
3559      <1>      mov     [csftdf_w_size], eax
3560      <1>
3561      <1>      mov     eax, [csftdf_filesize]
3562      <1>
3563      <1>      cmp     eax, 65536 ; 64KB ; small file
3564      <1>      jnb     short csftdf2_load_file ; do not display percentage
3565      <1>
3566      <1>      csftdf2_reset_wf_percent_ptr_chk_64k:
3567      <1>      mov     dl, 1 ; 25/03/2016
3568      <1>
3569      <1>      cmp     eax, 65536*4 ; 256KB
3570      <1>      jnb     short csftdf2_enable_percentage_display ; big file
3571      <1>
3572      <1>      ; 64-128KB file size for floppy disks
3573      <1>      cmp     byte [SourceFile_Drv], dl ; 1 ; read from floppy disk ?
3574      <1>      jna     short csftdf2_enable_percentage_display
3575      <1>
3576      <1>      cmp     byte [DestinationFile_Drv], dl ; 1 ; write to floppy disk ?
3577      <1>      ja      short csftdf2_load_file
3578      <1>
3579      <1>      csftdf2_enable_percentage_display:
3580      <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      <1>      mov     ah, 0Fh
3588      <1>      call    _int10h
3589      <1>      ; 13/05/2016
3590      <1>      mov     [csftdf_videopage], bh ; active video page
3591      <1>      mov     ah, 03h
3592      <1>      call    _int10h
3593      <1>      mov     [csftdf_cursorpos], dx
3594      <1>
3595      <1>      sub     eax, eax
3596      <1>      mov     [csftdf_rw_err], al ; 0
3597      <1>
3598      <1>      ; ///
3599      <1>      csftdf_sf_amb: ; 15/03/2016
3600      <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      <1>      call    allocate_memory_block
3613      <1>      jnc     short loc_check_sf_save_loading_parms
3614      <1>
3615      <1>      sub     eax, eax
3616      <1>      sub     ecx, ecx
3617      <1>
3618      <1>      loc_check_sf_save_loading_parms:
3619      <1>      mov     [csftdf_sf_mem_addr], eax ; loading address
3620      <1>      mov     [csftdf_sf_mem_bsize], ecx ; block size
3621      <1>      ; ///
3622      <1>      ; 19/03/2016
3623      <1>      mov     esi, [csftdf_sf_drv_dt] ; logical dos drv desc. tbl.
3624      <1>
3625      <1>      ; 17/03/2016
3626      <1>      or      eax, eax ; contiguous free memory block address
3627      <1>      jz      csftdf2_read_sf_cluster
3628      <1>
3629      <1>      ; 18/03/2016
3630      <1>      mov     ebx, [csftdf_sf_mem_addr] ; memory block address
3631      <1>
3632      <1>      cmp     byte [esi+LD_FATType], 0
3633      <1>      jna     csftdf2_load_fs_file
3634      <1>
3635      <1>      csftdf2_load_fat_file:
3636      <1>      push    ebx ; *
3637      <1>
3638      <1>      csftdf2_load_fat_file_next:
3639      <1>      mov     esi, msg_reading
3640      <1>      call    print_msg
3641      <1>
3642      <1>      cmp     byte [csftdf_percentage], 0
3643      <1>      jna     short csftdf2_load_fat_file_1
3644      <1>
3645      <1>      call    csftdf2_print_percentage ; 19/03/2016
3646      <1>
3647      <1>      csftdf2_load_fat_file_1:
3648      <1>      mov     esi, [csftdf_sf_drv_dt]

```

```

3649 0000B096 5B          <1>      pop     ebx ; *
3650                      <1>
3651                      <1> csftdf2_load_fat_file_2:
3652 0000B097 E8B8000000    <1>      call    csftdf2_read_fat_file_sectors ; 19/03/2016
3653 0000B09C 0F8250040000  <1>      jc      csftdf2_rw_error ; eocc! or disk error!
3654                      <1>
3655 0000B0A2 09D2          <1>      or      edx, edx ; edx > 0 -> EOF
3656 0000B0A4 7520          <1>      jnz     short csftdf2_load_fat_file_ok
3657                      <1>
3658 0000B0A6 803D[20450100]00 <1>      cmp     byte [csftdf_percentage], 0
3659 0000B0AD 76E8          <1>      jna     short csftdf2_load_fat_file_2
3660                      <1>
3661 0000B0AF 53            <1>      push    ebx ; *
3662                      <1>
3663                      <1>      ; Set cursor position
3664                      <1>      ; AH= 02h, BH= Page Number, DH= Row, DL= Column
3665 0000B0B0 8A3D[21450100] <1>      mov     bh, [csftdf_videopage]
3666 0000B0B6 668B15[22450100] <1>      mov     dx, [csftdf_cursorpos]
3667 0000B0BD B402          <1>      mov     ah, 2
3668 0000B0BF E8D663FFFF    <1>      call    _int10h
3669 0000B0C4 EBB2          <1>      jmp     short csftdf2_load_fat_file_next
3670                      <1>
3671                      <1> csftdf2_load_fat_file_ok:
3672 0000B0C6 803D[20450100]00 <1>      cmp     byte [csftdf_percentage], 0
3673 0000B0CD 0F8651020000  <1>      jna     csftdf2_save_file ; 25/03/2016
3674                      <1>
3675                      <1>      ; "Reading... 100%"
3676 0000B0D3 BF[0F010100]  <1>      mov     edi, percentagestr
3677 0000B0D8 B031          <1>      mov     al, '1'
3678 0000B0DA AA            <1>      stosb
3679 0000B0DB B030          <1>      mov     al, '0'
3680 0000B0DD AA            <1>      stosb
3681 0000B0DE AA            <1>      stosb
3682                      <1>
3683 0000B0DF 8A3D[21450100] <1>      mov     bh, [csftdf_videopage]
3684 0000B0E5 668B15[22450100] <1>      mov     dx, [csftdf_cursorpos]
3685 0000B0EC B402          <1>      mov     ah, 2
3686 0000B0EE E8A763FFFF    <1>      call    _int10h
3687                      <1>
3688 0000B0F3 BE[F7000100]  <1>      mov     esi, msg_reading
3689 0000B0F8 E8C0B1FFFF    <1>      call    print_msg
3690                      <1>
3691 0000B0FD BE[0F010100]  <1>      mov     esi, percentagestr
3692 0000B102 E8B6B1FFFF    <1>      call    print_msg
3693                      <1>
3694 0000B107 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 0000B10C B020          <1>      mov     al, 20h
3700 0000B10E BF[0F010100]  <1>      mov     edi, percentagestr
3701 0000B113 AA            <1>      stosb
3702 0000B114 AA            <1>      stosb
3703 0000B115 A1[18450100]  <1>      mov     eax, [csftdf_sf_rbytes]
3704 0000B11A BA64000000    <1>      mov     edx, 100
3705 0000B11F F7E2          <1>      mul     edx
3706 0000B121 8B0D[FC440100] <1>      mov     ecx, [csftdf_filesize]
3707 0000B127 F7F1          <1>      div     ecx
3708 0000B129 B10A          <1>      mov     cl, 10
3709 0000B12B F6F1          <1>      div     cl
3710 0000B12D 80C430        <1>      add     ah, '0'
3711 0000B130 8827          <1>      mov     [edi], ah
3712 0000B132 20C0          <1>      and     al, al
3713 0000B134 740A          <1>      jz      short csftdf2_print_percent_1
3714 0000B136 4F            <1>      dec     edi
3715 0000B137 6698          <1>      cbw
3716 0000B139 F6F1          <1>      div     cl
3717 0000B13B 80C430        <1>      add     ah, '0'
3718 0000B13E 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 0000B140 BE[0F010100]  <1>      mov     esi, percentagestr
3726                      <1>      ;call print_msg
3727                      <1>      ;retn
3728 0000B145 E973B1FFFF    <1>      jmp     print_msg
3729                      <1>
3730                      <1> csftdf2_read_file_sectors:
3731                      <1>      ; 19/03/2016
3732 0000B14A 807E0300        <1>      cmp     byte [esi+LD_FATType], 0
3733 0000B14E 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 0000B154 8B15[FC440100] <1>      mov     edx, [csftdf_filesize]
3745 0000B15A 2B15[18450100] <1>      sub     edx, [csftdf_sf_rbytes]
3746 0000B160 3B15[10450100] <1>      cmp     edx, [csftdf_r_size]
3747 0000B166 7306          <1>      jnb     short csftdf2_read_fat_file_secs_1
3748 0000B168 8915[10450100] <1>      mov     [csftdf_r_size], edx
3749                      <1>
3750                      <1> csftdf2_read_fat_file_secs_1:

```



```

3751 0000B16E A1[10450100] <1> mov     eax, [csftdf_r_size]
3752 0000B173 29D2 <1> sub     edx, edx
3753 0000B175 0FB74E11 <1> movzx   ecx, word [esi+LD_BPB+BytesPerSec]
3754 0000B179 01C8 <1> add     eax, ecx
3755 0000B17B 48 <1> dec     eax
3756 0000B17C F7F1 <1> div     ecx
3757 0000B17E 89C1 <1> mov     ecx, eax ; sector count
3758 0000B180 A1[08450100] <1> mov     eax, [csftdf_sf_cluster]
3759 <1>
3760 <1> ; EBX = memory block address (current)
3761 <1>
3762 0000B185 E821090000 <1> call    read_fat_file_sectors
3763 0000B18A 7235 <1> jc      short csftdf2_read_fat_file_secs_3
3764 <1>
3765 <1> ; EBX = next memory address
3766 <1>
3767 0000B18C A1[18450100] <1> mov     eax, [csftdf_sf_rbytes]
3768 0000B191 0305[10450100] <1> add     eax, [csftdf_r_size]
3769 0000B197 8B15[FC440100] <1> mov     edx, [csftdf_filesize]
3770 0000B19D 39D0 <1> cmp     eax, edx
3771 0000B19F 7320 <1> jnb     short csftdf2_read_fat_file_secs_3 ; edx > 0
3772 0000B1A1 A3[18450100] <1> mov     [csftdf_sf_rbytes], eax
3773 <1>
3774 0000B1A6 53 <1> push    ebx ; *
3775 <1> ; get next cluster (csftdf_r_size! bytes)
3776 0000B1A7 A1[08450100] <1> mov     eax, [csftdf_sf_cluster]
3777 0000B1AC E8CC060000 <1> call    get_next_cluster
3778 0000B1B1 5B <1> pop     ebx ; *
3779 0000B1B2 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 0000B1B4 B811000000 <1> mov     eax, 17 ; Read error !
3784 0000B1B9 C3 <1> retn
3785 <1>
3786 <1> csftdf2_read_fat_file_secs_2:
3787 0000B1BA 29D2 <1> sub     edx, edx ; 0
3788 0000B1BC A3[08450100] <1> mov     [csftdf_sf_cluster], eax ; next cluster
3789 <1>
3790 <1> csftdf2_read_fat_file_secs_3:
3791 0000B1C1 C3 <1> retn
3792 <1>
3793 <1> csftdf2_read_sf_cluster:
3794 <1> ; 19/03/2016
3795 0000B1C2 BB00000700 <1> mov     ebx, Cluster_Buffer ; buffer address (64KB)
3796 <1>
3797 0000B1C7 803D[20450100]00 <1> cmp     byte [csftdf_percentage], 0
3798 0000B1CE 760D <1> jna     short csftdf2_read_sf_clust_2
3799 <1>
3800 0000B1D0 53 <1> push    ebx ; *
3801 <1>
3802 <1> csftdf2_read_sf_clust_next:
3803 0000B1D1 E836FFFFFF <1> call    csftdf2_print_percentage
3804 <1>
3805 <1> csftdf2_read_sf_clust_0:
3806 0000B1D6 8B35[24450100] <1> mov     esi, [csftdf_sf_drv_dt]
3807 <1> csftdf2_read_sf_clust_1:
3808 0000B1DC 5B <1> pop     ebx ; *
3809 <1>
3810 <1> csftdf2_read_sf_clust_2:
3811 0000B1DD 89DA <1> mov     edx, ebx
3812 0000B1DF 0315[10450100] <1> add     edx, [csftdf_r_size]
3813 0000B1E5 81FA00000800 <1> cmp     edx, Cluster_Buffer + 65536
3814 0000B1EB 772F <1> ja      short csftdf2_write_df_cluster
3815 <1>
3816 0000B1ED E858FFFFFF <1> call    csftdf2_read_file_sectors ; 19/03/2016
3817 0000B1F2 0F8280020000 <1> jc      csftdf2_save_fat_file_err2 ; eocc! or disk error!
3818 <1>
3819 0000B1F8 09D2 <1> or      edx, edx ; edx > 0 -> EOF
3820 0000B1FA 7520 <1> jnz     short csftdf2_write_df_cluster
3821 <1>
3822 0000B1FC 803D[20450100]00 <1> cmp     byte [csftdf_percentage], 0
3823 0000B203 76D8 <1> jna     short csftdf2_read_sf_clust_2
3824 <1>
3825 0000B205 53 <1> push    ebx ; *
3826 <1>
3827 <1> ; Set cursor position
3828 <1> ; AH= 02h, BH= Page Number, DH= Row, DL= Column
3829 0000B206 8A3D[21450100] <1> mov     bh, [csftdf_videopage]
3830 0000B20C 668B15[22450100] <1> mov     dx, [csftdf_cursorpos]
3831 0000B213 B402 <1> mov     ah, 2
3832 0000B215 E88062FFFF <1> call    _int10h
3833 0000B21A EBB5 <1> jmp     short csftdf2_read_sf_clust_next
3834 <1>
3835 <1> csftdf2_write_df_cluster:
3836 <1> ; 19/03/2016
3837 0000B21C 8B35[28450100] <1> mov     esi, [csftdf_df_drv_dt]
3838 0000B222 BB00000700 <1> mov     ebx, Cluster_Buffer ; buffer address (64KB)
3839 <1>
3840 <1> csftdf2_write_df_clust_next:
3841 0000B227 E855000000 <1> call    csftdf2_write_file_sectors ; 19/03/2016
3842 0000B22C 0F8246020000 <1> jc      csftdf2_save_fat_file_err2 ; eocc! or disk error!
3843 <1>
3844 0000B232 09D2 <1> or      edx, edx ; edx > 0 -> EOF
3845 0000B234 750A <1> jnz     short csftdf2_rw_f_clust_ok
3846 <1>
3847 0000B236 81FB00000800 <1> cmp     ebx, Cluster_Buffer + 65536
3848 0000B23C 72E9 <1> jb      short csftdf2_write_df_clust_next
3849 <1>
3850 0000B23E EB82 <1> jmp     short csftdf2_read_sf_cluster
3851 <1>
3852 <1> csftdf2_rw_f_clust_ok:

```

```

3853 0000B240 803D[20450100]00 <1>      cmp     byte [csftdf_percentage], 0
3854 0000B247 0F86B2010000 <1>      jna     csftdf2_save_fat_file_4 ; 25/03/2016
3855 <1>
3856 <1>      ; "100%"
3857 0000B24D BF[0F010100] <1>      mov     edi, percentagestr
3858 0000B252 B031 <1>      mov     al, '1'
3859 0000B254 AA <1>      stosb
3860 0000B255 B030 <1>      mov     al, '0'
3861 0000B257 AA <1>      stosb
3862 0000B258 AA <1>      stosb
3863 <1>
3864 0000B259 8A3D[21450100] <1>      mov     bh, [csftdf_videopage]
3865 0000B25F 668B15[22450100] <1>      mov     dx, [csftdf_cursorpos]
3866 0000B266 B402 <1>      mov     ah, 2
3867 0000B268 E82D62FFFF <1>      call    _int10h
3868 <1>
3869 0000B26D BE[0F010100] <1>      mov     esi, percentagestr
3870 0000B272 E846B0FFFF <1>      call    print_msg
3871 <1>
3872 0000B277 E983010000 <1>      jmp     csftdf2_save_fat_file_4
3873 <1>
3874 <1> csftdf2_load_fs_file:
3875 <1>      ; temporary - 18/03/2016
3876 0000B27C E96F020000 <1>      jmp     csftdf2_read_error
3877 <1>
3878 <1> csftdf2_write_file_sectors:
3879 <1>      ; 19/03/2016
3880 0000B281 807E0300 <1>      cmp     byte [esi+LD_FATType], 0
3881 0000B285 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 0000B28B 8B15[FC440100] <1>      mov     edx, [csftdf_filesize]
3893 0000B291 2B15[1C450100] <1>      sub     edx, [csftdf_df_wbytes]
3894 0000B297 3B15[14450100] <1>      cmp     edx, [csftdf_w_size]
3895 0000B29D 7306 <1>      jnb     short csftdf2_write_fat_file_secs_1
3896 0000B29F 8915[14450100] <1>      mov     [csftdf_w_size], edx
3897 <1>
3898 <1> csftdf2_write_fat_file_secs_1:
3899 0000B2A5 A1[14450100] <1>      mov     eax, [csftdf_w_size]
3900 0000B2AA 29D2 <1>      sub     edx, edx
3901 0000B2AC 0FB74E11 <1>      movzx   ecx, word [esi+LD_BPB+BytesPerSec]
3902 0000B2B0 01C8 <1>      add     eax, ecx
3903 0000B2B2 48 <1>      dec     eax
3904 0000B2B3 F7F1 <1>      div     ecx
3905 0000B2B5 89C1 <1>      mov     ecx, eax ; sector count
3906 0000B2B7 A1[0C450100] <1>      mov     eax, [csftdf_df_cluster]
3907 <1>
3908 <1>      ; EBX = memory block address (current)
3909 <1>
3910 0000B2BC E8A20F0000 <1>      call    write_fat_file_sectors
3911 0000B2C1 7259 <1>      jc     short csftdf2_write_fat_file_secs_4
3912 <1>
3913 <1>      ; EBX = next memory address
3914 <1>
3915 0000B2C3 A1[1C450100] <1>      mov     eax, [csftdf_df_wbytes]
3916 0000B2C8 0305[14450100] <1>      add     eax, [csftdf_w_size]
3917 0000B2CE 8B15[FC440100] <1>      mov     edx, [csftdf_filesize]
3918 0000B2D4 39D0 <1>      cmp     eax, edx
3919 0000B2D6 7344 <1>      jnb     short csftdf2_write_fat_file_secs_4
3920 0000B2D8 A3[1C450100] <1>      mov     [csftdf_df_wbytes], eax
3921 <1>      ;
3922 0000B2DD A3[BE440100] <1>      mov     [DestinationFile_DirEntry+DirEntry_FileSize], eax
3923 <1>
3924 0000B2E2 53 <1>      push    ebx ; *
3925 <1>
3926 0000B2E3 803D[FA440100]01 <1>      cmp     byte [DestinationFileFound], 1
3927 0000B2EA 7210 <1>      jnb     short csftdf2_write_fat_file_secs_2
3928 <1>
3929 <1>      ; get next cluster (csftdf_w_size! bytes)
3930 0000B2EC A1[0C450100] <1>      mov     eax, [csftdf_df_cluster]
3931 0000B2F1 E887050000 <1>      call    get_next_cluster
3932 0000B2F6 731C <1>      jnc     short csftdf2_write_fat_file_secs_3
3933 <1>
3934 0000B2F8 21C0 <1>      and     eax, eax ; end of cluster chain!?
3935 0000B2FA 7521 <1>      jnz     short csftdf2_write_fat_file_secs_5 ; disk error !
3936 <1>
3937 <1> csftdf2_write_fat_file_secs_2:
3938 0000B2FC A1[0C450100] <1>      mov     eax, [csftdf_df_cluster] ; last cluster
3939 0000B301 E8800E0000 <1>      call    add_new_cluster
3940 0000B306 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 0000B308 8B15[14450100] <1>      mov     edx, [csftdf_w_size] ; bytes per cluster
3948 0000B30E 0115[BE440100] <1>      add     [DestinationFile_DirEntry+DirEntry_FileSize], edx
3949 <1>
3950 <1> csftdf2_write_fat_file_secs_3:
3951 0000B314 5B <1>      pop     ebx ; *
3952 0000B315 29D2 <1>      sub     edx, edx ; 0
3953 0000B317 A3[0C450100] <1>      mov     [csftdf_df_cluster], eax ; next cluster
3954 <1>

```

```

3955 <1> csftdf2_write_fat_file_secs_4:
3956 <1>     retn
3957 <1>
3958 <1> csftdf2_write_fat_file_secs_5:
3959 <1>     pop     ebx ; *
3960 <1>     ; 16/10/2016 (1Dh -> 18)
3961 <1>     mov     eax, 18 ; Write error !
3962 <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 <1>     mov     esi, [csftdf_df_drv_dt] ; logical dos drv desc. tbl.
3969 <1>
3970 <1>     mov     ebx, [csftdf_sf_mem_addr] ; memory block address
3971 <1>
3972 <1>     cmp     byte [esi+LD_FATType], 0
3973 <1>     jna     csftdf2_save_fs_file
3974 <1>
3975 <1> csftdf2_save_fat_file:
3976 <1>     push    ebx; *
3977 <1>
3978 <1>     cmp     byte [csftdf_percentage], 0
3979 <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 <1>     mov     bh, [csftdf_videopage]
3984 <1>     mov     dx, [csftdf_cursorpos]
3985 <1>     mov     ah, 2
3986 <1>     call    _int10h
3987 <1>
3988 <1>     mov     esi, msg_writing
3989 <1>     call    print_msg
3990 <1>
3991 <1> csftdf2_save_fat_file_next:
3992 <1>     mov     esi, [csftdf_df_drv_dt] ; 25/03/2016
3993 <1>
3994 <1> csftdf2_save_fat_file_0:
3995 <1>     pop     ebx ; *
3996 <1>
3997 <1> csftdf2_save_fat_file_1:
3998 <1>     call    csftdf2_write_file_sectors ; 19/03/2016
3999 <1>     jc     csftdf2_rw_error ; eocc! or disk error!
4000 <1>
4001 <1>     or      edx, edx ; edx > 0 -> EOF
4002 <1>     jnz     short csftdf2_save_fat_file_3 ; 25/03/2016
4003 <1>
4004 <1>     cmp     byte [csftdf_percentage], 0
4005 <1>     jna     short csftdf2_save_fat_file_1
4006 <1>
4007 <1>     mov     al, 20h
4008 <1>     mov     edi, percentagestr
4009 <1>     stosb
4010 <1>     stosb
4011 <1>     mov     eax, [csftdf_df_wbytes]
4012 <1>     mov     edx, 100
4013 <1>     mul     edx
4014 <1>     mov     ecx, [csftdf_filesize]
4015 <1>     div     ecx
4016 <1>     mov     cl, 10
4017 <1>     div     cl
4018 <1>     add     ah, '0'
4019 <1>     mov     [edi], ah
4020 <1>     and     al, al
4021 <1>     jz      short csftdf2_save_fat_file_2
4022 <1>     dec     edi
4023 <1>     cbw
4024 <1>     div     cl
4025 <1>     add     ah, '0'
4026 <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 <1>     push    ebx ; *
4034 <1>
4035 <1>     call    csftdf2_print_wr_percentage ; 25/03/2016
4036 <1>
4037 <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 <1>     mov     bh, [csftdf_videopage]
4043 <1>     mov     dx, [csftdf_cursorpos]
4044 <1>     mov     ah, 2
4045 <1>     call    _int10h
4046 <1>
4047 <1>     mov     esi, msg_writing
4048 <1>     call    print_msg
4049 <1>
4050 <1>     mov     esi, percentagestr
4051 <1>     ;call   print_msg
4052 <1>     ;retn
4053 <1>     jmp     print_msg
4054 <1>
4055 <1> csftdf2_save_fat_file_3:
4056 <1>     cmp     byte [csftdf_percentage], 0

```

```

4057 0000B3EC 7611      <1>          jna      csftdf2_save_fat_file_4 ; 25/03/2016
4058                      <1>
4059                      <1>          ; "100%"
4060 0000B3EE BF[0F010100] <1>          mov     edi, percentagestr
4061 0000B3F3 B031      <1>          mov     al, '1'
4062 0000B3F5 AA        <1>          stosb
4063 0000B3F6 B030      <1>          mov     al, '0'
4064 0000B3F8 AA        <1>          stosb
4065 0000B3F9 AA        <1>          stosb
4066                      <1>
4067 0000B3FA E8BEFFFFFF <1>          call    csftdf2_print_wr_percentage
4068                      <1>
4069                      <1> csftdf2_save_fat_file_4:
4070 0000B3FF 803D[FA440100]00 <1>          cmp     byte [DestinationFileFound], 0
4071 0000B406 7647      <1>          jna     short csftdf2_save_fat_file_6
4072                      <1>
4073 0000B408 8B35[28450100] <1>          mov     esi, [csftdf_df_drv_dt] ; 31/03/2016
4074                      <1>
4075 0000B40E A1[0C450100] <1>          mov     eax, [csftdf_df_cluster] ; last cluster
4076 0000B413 E865040000 <1>          call    get_next_cluster
4077 0000B418 7235      <1>          jc     short csftdf2_save_fat_file_6 ; eocc! or disk error!
4078                      <1>
4079 0000B41A A1[0C450100] <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 ; 0FFFFFFFh
4083                      <1>          ;shr     ecx, 4 ; 28 bit ; 0FFFFFFFh
4084 0000B41F B9FFFFFF0F <1>          mov     ecx, 0FFFFFFFh
4085 0000B424 E87E070000 <1>          call    update_cluster
4086 0000B429 7224      <1>          jc     short csftdf2_save_fat_file_6 ; really last cluster!?
4087                      <1>
4088 0000B42B A3[0C450100] <1>          mov     [csftdf_df_cluster], eax ; next cluster
4089                      <1>
4090                      <1>          ; byte [FAT_BuffValidData] = 2
4091 0000B430 E82F0A0000 <1>          call    save_fat_buffer
4092 0000B435 730E      <1>          jnc     short csftdf2_save_fat_file_5
4093                      <1>
4094 0000B437 8B15[FC440100] <1>          mov     edx, [csftdf_filesize]
4095 0000B43D 8915[BE440100] <1>          mov     [DestinationFile_DirEntry+DirEntry_FileSize], edx
4096 0000B443 EB58      <1>          jmp     short csftdf2_save_fat_file_err3
4097                      <1>
4098                      <1> csftdf2_save_fat_file_5:
4099 0000B445 A1[0C450100] <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 0000B44A E8580C0000 <1>          call    truncate_cluster_chain
4104                      <1>
4105                      <1> csftdf2_save_fat_file_6:
4106                      <1>          ; 28/03/2016
4107 0000B44F BE[2D440100] <1>          mov     esi, SourceFile_DirEntry+DirEntry_Attr ; +11 to + 18
4108 0000B454 BF[AD440100] <1>          mov     edi, DestinationFile_DirEntry+DirEntry_Attr ; +11 to + 18
4109 0000B459 A4        <1>          movsb ; +11
4110 0000B45A A5        <1>          movsd ; +12 .. +15
4111 0000B45B 66A5      <1>          movsw ; +16 .. +17
4112                      <1>          ; + 18
4113 0000B45D 83C604    <1>          add     esi, 4
4114 0000B460 83C704    <1>          add     edi, 4
4115 0000B463 A5        <1>          movsd ; DirEntry_WrtTime ; +22 .. +25
4116                      <1>
4117 0000B464 8B15[FC440100] <1>          mov     edx, [csftdf_filesize]
4118 0000B46A 8915[BE440100] <1>          mov     [DestinationFile_DirEntry+DirEntry_FileSize], edx
4119                      <1>
4120 0000B470 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 0000B475 EB4D      <1>          jmp     short csftdf2_save_fat_file_7
4124                      <1>
4125                      <1> csftdf2_save_fat_file_err1:
4126 0000B477 5B        <1>          pop     ebx ; *
4127                      <1> csftdf2_save_fat_file_err2:
4128 0000B478 A1[1C450100] <1>          mov     eax, [csftdf_df_wbytes]
4129 0000B47D 8B15[BE440100] <1>          mov     edx, [DestinationFile_DirEntry+DirEntry_FileSize]
4130 0000B483 39C2      <1>          cmp     edx, eax
4131 0000B485 7616      <1>          jna     short csftdf2_save_fat_file_err3
4132 0000B487 A1[0C450100] <1>          mov     eax, [csftdf_df_cluster] ; last (empty) cluster
4133                      <1>          ; ESI = Logical dos drive description table address
4134 0000B48C E8160C0000 <1>          call    truncate_cluster_chain
4135 0000B491 720A      <1>          jc     short csftdf2_save_fat_file_err3
4136 0000B493 A1[1C450100] <1>          mov     eax, [csftdf_df_wbytes]
4137 0000B498 A3[BE440100] <1>          mov     [DestinationFile_DirEntry+DirEntry_FileSize], eax
4138                      <1> csftdf2_save_fat_file_err3:
4139 0000B49D 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 0000B4A2 C605[AF440100]00 <1>          mov     byte [DestinationFile_DirEntry+DirEntry_CrtTimeTenth], 0
4143 0000B4A9 66A3[B0440100] <1>          mov     [DestinationFile_DirEntry+DirEntry_CrtTime], ax
4144 0000B4AF 668915[B2440100] <1>          mov     [DestinationFile_DirEntry+DirEntry_CrtDate], dx
4145 0000B4B6 66A3[B8440100] <1>          mov     [DestinationFile_DirEntry+DirEntry_WrtTime], ax
4146 0000B4BC 668915[BA440100] <1>          mov     [DestinationFile_DirEntry+DirEntry_WrtDate], dx
4147 0000B4C3 F9        <1>          stc
4148                      <1> csftdf2_save_fat_file_7:
4149 0000B4C4 9C        <1>          pushf
4150 0000B4C5 668915[B4440100] <1>          mov     [DestinationFile_DirEntry+DirEntry_LastAccDate], dx
4151 0000B4CC BE[A2440100] <1>          mov     esi, DestinationFile_DirEntry
4152 0000B4D1 BF00000800 <1>          mov     edi, Directory_Buffer
4153 0000B4D6 0FB70D[CA440100] <1>          movzx   ecx, word [DestinationFile_DirEntryNumber] ; (<2048)
4154 0000B4DD 66C1E105 <1>          shl     cx, 5 ; 32 * directory entry number
4155 0000B4E1 01CF      <1>          add     edi, ecx
4156                      <1>          ;mov     ecx, 8
4157 0000B4E3 66B90800 <1>          mov     cx, 8
4158 0000B4E7 F3A5      <1>          rep     movsd

```



```

4159 0000B4E9 9D          <1>      popf
4160 0000B4EA 730B        <1>      jnc     short csftdf2_write_file_OK
4161                                <1>
4162                                <1> csftdf2_write_error:
4163                                <1>      ; 18/03/2016
4164 0000B4EC B01D        <1>      mov     al, 1Dh ; write error
4165 0000B4EE EB02        <1>      jmp     short csftdf2_rw_error
4166                                <1>
4167                                <1>      ; 16/03/2016
4168                                <1> csftdf2_read_error:
4169 0000B4F0 B011        <1>      mov     al, 17 ; ; Drive not ready or read error!
4170                                <1> csftdf2_rw_error:
4171 0000B4F2 A2[F9440100] <1>      mov     [csftdf_rw_err], al
4172                                <1>
4173                                <1> csftdf2_write_file_OK:
4174                                <1>      ; 18/03/2016
4175 0000B4F7 C605[10410100]02 <1>      mov     byte [DirBuff_ValidData], 2
4176 0000B4FE E8C7F0FFFF <1>      call    save_directory_buffer
4177                                <1>
4178                                <1>      ; Update last modification date&time of destination
4179                                <1>      ; file's (parent) directory
4180 0000B503 E85DF1FFFF <1>      call    update_parent_dir_lmdt
4181                                <1>      ;
4182 0000B508 A1[00450100] <1>      mov     eax, [csftdf_sf_mem_addr] ; start address
4183                                <1>
4184 0000B50D 21C0        <1>      and     eax, eax
4185 0000B50F 750E        <1>      jnz     short csftdf2_dealloc_mblock
4186                                <1>
4187 0000B511 88C5        <1>      mov     ch, al ; 0 (Cluster r/w, not full loading)
4188                                <1> csftdf2_dealloc_retn:
4189 0000B513 8A0D[F9440100] <1>      mov     cl, [csftdf_rw_err]
4190 0000B519 A1[0C450100] <1>      mov     eax, [csftdf_df_cluster]
4191 0000B51E C3          <1>      retn
4192                                <1>
4193                                <1> csftdf2_dealloc_mblock:
4194 0000B51F 8B0D[04450100] <1>      mov     ecx, [csftdf_sf_mem_bsize] ; block size
4195 0000B525 E8FAA0FFFF <1>      call    deallocate_memory_block
4196 0000B52A B5FF        <1>      mov     ch, 0FFh ; (File was full loaded at memory)
4197 0000B52C 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 0000B52E B812000000 <1>      mov     eax, 18 ; write error
4203 0000B533 F9          <1>      stc
4204 0000B534 C3          <1>      retn
4205                                <1>
4206                                <1> create_file:
4207                                <1>      ; 16/10/2016
4208                                <1>      ; 24/03/2016, 31/03/2016
4209                                <1>      ; 20/03/2016, 21/03/2016, 23/03/2016
4210                                <1>      ; 19/03/2016 (TRDOS 396 = TRDOS v2.0)
4211                                <1>      ; 03/09/2011 (FILE.ASM, 'proc_create_file')
4212                                <1>      ; 09/08/2010
4213                                <1>      ;
4214                                <1>      ; INPUT ->
4215                                <1>      ;      EAX = File Size
4216                                <1>      ;      ESI = ASCIIZ File Name
4217                                <1>      ;      CL = File Attributes
4218                                <1>      ;      EBX = FFFFFFFFh -> create empty file
4219                                <1>      ;      (only for FAT fs)
4220                                <1>      ; OUTPUT ->
4221                                <1>      ;      CF = 0 ->
4222                                <1>      ;      EAX = New file's first cluster
4223                                <1>      ;      ESI = Logical Dos Drv Descr. Table Addr.
4224                                <1>      ;      EBX = offset CreateFile_Size
4225                                <1>      ;      ECX = Sectors per cluster (<256)
4226                                <1>      ;      EDX = Directory entry index/number (<65536)
4227                                <1>      ;      CF = 1 -> error code in AL
4228                                <1>
4229                                <1> ; test cl, 18h (directory or volume name)
4230                                <1> ; jnz short loc_createfile_access_denied
4231 0000B535 80E107        <1>      and     cl, 07h ; S, H, R
4232 0000B538 880D[48450100] <1>      mov     [createfile_attr], cl
4233                                <1>
4234 0000B53E 89D9        <1>      mov     ecx, ebx
4235 0000B540 89F3        <1>      mov     ebx, esi ; ASCIIZ File Name address
4236 0000B542 29D2        <1>      sub     edx, edx
4237 0000B544 8A35[E6380100] <1>      mov     dh, [Current_Drv]
4238 0000B54A BE00010900 <1>      mov     esi, Logical_DOSDisks
4239 0000B54F 01D6        <1>      add     esi, edx
4240                                <1>
4241 0000B551 8815[53450100] <1>      mov     [createfile_UpdatePDir], dl ; 0 ; 31/03/2016
4242                                <1>
4243                                <1>      ; LD_DiskType = 0 for write protection (read only)
4244 0000B557 807E0101 <1>      cmp     byte [esi+LD_DiskType], 1 ; 0 = Invalid
4245 0000B55B 730A        <1>      jnb     short loc_createfile_check_file_sytem
4246                                <1>      ; 16/10/2016 (TRDOS Error code: 30, disk write protected)
4247 0000B55D B81E000000 <1>      mov     eax, 30 ; 13h, MSDOS err : Disk write-protected
4248 0000B562 66BA0000 <1>      mov     dx, 0
4249                                <1>      ; err retn: EDX = 0, EBX = File name offset
4250                                <1>      ; ESI -> Dos drive description table address
4251 0000B566 C3          <1>      retn
4252                                <1>
4253                                <1> ;loc_createfile_access_denied:
4254                                <1> ; mov     eax, 05h ; access denied (invalid attributes input)
4255                                <1> ; stc
4256                                <1> ; retn
4257                                <1>
4258                                <1> loc_createfile_check_file_sytem:
4259 0000B567 807E0301 <1>      cmp     byte [esi+LD_FATType], 1
4260 0000B56B 730A        <1>      jnb     short loc_createfile_chk_empty_FAT_file_sign1

```

```

4261                                     <1>
4262 0000B56D A3[34450100]             <1>      mov     [createfile_size], eax
4263                                     <1>      ; ESI = Logical Dos Drive Description Table address
4264                                     <1>      ; EBX = ASCIIIZ File Name address
4265 0000B572 E9FE020000             <1>      jmp     create_fs_file
4266                                     <1>
4267                                     <1> loc_createfile_chk_empty_FAT_file_sign1:
4268                                     <1>      ; ECX = FFFFFFFFh -> create empty file if drive has FAT fs
4269 0000B577 41                     <1>      inc     ecx
4270 0000B578 7506                   <1>      jnz     short loc_createfile_chk_empty_FAT_file_sign2
4271 0000B57A 890D[34450100]         <1>      mov     [createfile_size], ecx ; 0 ; empty file
4272                                     <1>
4273                                     <1> loc_createfile_chk_empty_FAT_file_sign2:
4274                                     <1>      ; 23/03/2016
4275 0000B580 668B4E11             <1>      mov     cx, [esi+LD_BPB+BytesPerSec]
4276 0000B584 66890D[50450100]       <1>      mov     [createfile_BytesPerSec], cx
4277                                     <1>
4278                                     <1>      ; EBX = ASCIIIZ File Name address
4279 0000B58B 0FB65613             <1>      movzx   edx, byte [esi+LD_BPB+SecPerClust]
4280 0000B58F 8815[49450100]         <1>      mov     [createfile_SecPerClust], dl
4281 0000B595 8B4E74               <1>      mov     ecx, [esi+LD_FreeSectors]
4282 0000B598 39D1                 <1>      cmp     ecx, edx ; byte [createfile_SecPerClust]
4283 0000B59A 7306                 <1>      jnb     short loc_create_fat_file
4284                                     <1>
4285                                     <1> loc_createfile_insufficient_disk_space:
4286 0000B59C B827000000           <1>      mov     eax, 27h
4287                                     <1> loc_createfile_gffc_retn:
4288 0000B5A1 C3                   <1>      retn
4289                                     <1>
4290                                     <1> loc_create_fat_file:
4291 0000B5A2 891D[2C450100]         <1>      mov     [createfile_Name_Offset], ebx
4292 0000B5A8 890D[30450100]         <1>      mov     [createfile_FreeSectors], ecx
4293                                     <1>
4294                                     <1> loc_createfile_gffc_1:
4295 0000B5AE E821050000           <1>      call    get_first_free_cluster
4296 0000B5B3 72EC                 <1>      jc      short loc_createfile_gffc_retn
4297                                     <1>
4298 0000B5B5 A3[38450100]         <1>      mov     [createfile_FFCluster], eax
4299                                     <1>
4300                                     <1> loc_createfile_locate_ffc_on_directory:
4301                                     <1>      ; Current directory fcluster <> Directory buffer cluster
4302                                     <1>      ; Current directory will be reloaded by
4303                                     <1>      ; 'locate_current_dir_file' procedure
4304                                     <1>      ;
4305                                     <1>      ; ESI = Logical Dos Drv Desc. Table Address
4306 0000B5BA 56                     <1>      push    esi ; *
4307 0000B5BB 31C0                 <1>      xor     eax, eax
4308                                     <1>
4309 0000B5BD A3[06410100]         <1>      mov     dword [FAT_ClusterCounter], eax ; 0
4310                                     <1>      ; 21/03/2016
4311 0000B5C2 A2[52450100]         <1>      mov     byte [createfile_wfc], al ; 0
4312                                     <1>
4313 0000B5C7 89C1                 <1>      mov     ecx, eax
4314 0000B5C9 6649                 <1>      dec     cx ; FFFFh
4315                                     <1>      ; CX = FFFFh -> find first deleted or free entry
4316                                     <1>      ; ESI would be ASCIIIZ filename address if the call
4317                                     <1>      ; would not be for first free or deleted dir entry
4318 0000B5CB E8D6E7FFFF           <1>      call    locate_current_dir_file
4319 0000B5D0 0F83EE000000         <1>      jnc     loc_createfile_set_ff_dir_entry
4320 0000B5D6 5E                     <1>      pop     esi ; *
4321                                     <1>      ; ESI = Logical DOS Drv. Description Table Address
4322 0000B5D7 83F802               <1>      cmp     eax, 2
4323 0000B5DA 7402                 <1>      je      short loc_createfile_add_new_cluster
4324                                     <1> loc_createfile_locate_file_stc_retn:
4325 0000B5DC F9                     <1>      stc
4326 0000B5DD C3                   <1>      retn
4327                                     <1>
4328                                     <1> loc_createfile_add_new_cluster:
4329 0000B5DE 803D[E5380100]02       <1>      cmp     byte [Current_FATType], 2
4330                                     <1>      ;cmp     byte [esi+LD_FATType], 2
4331 0000B5E5 770C                 <1>      ja      short loc_createfile_add_new_cluster_check_fsc
4332 0000B5E7 803D[E4380100]01       <1>      cmp     byte [Current_Dir_Level], 1
4333                                     <1>      ;cmp     byte [esi+LD_CDirLevel], 1
4334 0000B5EE 7303                 <1>      jnb     short loc_createfile_add_new_cluster_check_fsc
4335                                     <1>
4336                                     <1>      ;mov     eax, 12
4337 0000B5F0 B00C                 <1>      mov     al, 12 ; No more files
4338                                     <1>
4339                                     <1> loc_createfile_anc_retn:
4340 0000B5F2 C3                   <1>      retn
4341                                     <1>
4342                                     <1> loc_createfile_add_new_cluster_check_fsc:
4343 0000B5F3 8B0D[30450100]         <1>      mov     ecx, [createfile_FreeSectors]
4344 0000B5F9 0FB605[49450100]       <1>      movzx   eax, byte [createfile_SecPerClust]
4345 0000B600 66D1E0               <1>      shl     ax, 1 ; AX = 2 * AX
4346 0000B603 39C1                 <1>      cmp     ecx, eax
4347 0000B605 7295                 <1>      jb      short loc_createfile_insufficient_disk_space
4348                                     <1>
4349                                     <1> loc_createfile_add_new_subdir_cluster:
4350 0000B607 8B15[15410100]         <1>      mov     edx, [DirBuff_Cluster]
4351 0000B60D 8915[3C450100]         <1>      mov     [createfile_LastDirCluster], edx
4352                                     <1>
4353 0000B613 A1[38450100]         <1>      mov     eax, [createfile_FFCluster]
4354 0000B618 E846040000           <1>      call    load_FAT_sub_directory
4355 0000B61D 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 0000B61F 0FB705[50450100]       <1>      movzx   eax, word [createfile_BytesPerSec] ; 23/03/2016
4360 0000B626 F7E1                 <1>      mul     ecx ; ecx = directory buffer sector count
4361 0000B628 89C1                 <1>      mov     ecx, eax
4362 0000B62A C1E902               <1>      shr     ecx, 2 ; dword count

```

```

4363 0000B62D 29C0      <1>      sub     eax, eax ; 0
4364 0000B62F F3AB      <1>      rep     stosd
4365                      <1>      ;
4366 0000B631 C605[10410100]02 <1>      mov     byte [DirBuff_ValidData], 2
4367 0000B638 E88DEFFFFFFF <1>      call    save_directory_buffer
4368 0000B63D 72B3      <1>      jc      short loc_createfile_anc_retn
4369                      <1>
4370                      <1> loc_createfile_save_added_subdir_cluster:
4371 0000B63F A1[3C450100]    <1>      mov     eax, [createfile_LastDirCluster]
4372 0000B644 8B0D[38450100] <1>      mov     ecx, [createfile_FFCluster]
4373 0000B64A E858050000    <1>      call    update_cluster
4374 0000B64F 7304      <1>      jnc     short loc_createfile_save_fat_buffer_0
4375 0000B651 09C0      <1>      or      eax, eax ; EAX = 0 -> cluster value is 0 or eocc
4376 0000B653 751A      <1>      jnz     short loc_createfile_save_fat_buffer_stc_retn
4377                      <1>
4378                      <1> loc_createfile_save_fat_buffer_0:
4379 0000B655 A1[38450100]    <1>      mov     eax, [createfile_FFCluster]
4380 0000B65A A3[3C450100]    <1>      mov     [createfile_LastDirCluster], eax
4381 0000B65F B9FFFFFFF0F      <1>      mov     ecx, 0FFFFFFFh ; 28 bit
4382 0000B664 E83E050000    <1>      call    update_cluster
4383 0000B669 7306      <1>      jnc     short loc_createfile_save_fat_buffer_1
4384 0000B66B 09C0      <1>      or      eax, eax ; Was it free cluster
4385 0000B66D 7402      <1>      jz      short loc_createfile_save_fat_buffer_1
4386                      <1>
4387                      <1> loc_createfile_save_fat_buffer_stc_retn:
4388 0000B66F F9          <1>      stc
4389                      <1> loc_createfile_save_fat_buffer_retn:
4390                      <1> loc_createfile_gffc_2_stc_retn:
4391 0000B670 C3          <1>      retn
4392                      <1>
4393                      <1> loc_createfile_save_fat_buffer_1:
4394                      <1>      ; byte [FAT_BuffValidData] = 2
4395 0000B671 E8EE070000    <1>      call    save_fat_buffer
4396 0000B676 72F8      <1>      jc      short loc_createfile_save_fat_buffer_retn
4397                      <1>
4398 0000B678 803D[06410100]01 <1>      cmp     byte [FAT_ClusterCounter], 1
4399 0000B67F 7222      <1>      jb      short loc_createfile_save_fat_buffer_2
4400                      <1>
4401                      <1>      ; ESI = Logical DOS Drive Description Table address
4402 0000B681 A1[06410100]    <1>      mov     eax, [FAT_ClusterCounter]
4403                      <1>
4404 0000B686 C605[06410100]00 <1>      mov     byte [FAT_ClusterCounter], 0 ; 21/03/2016
4405                      <1>
4406 0000B68D 66BB01FF      <1>      mov     bx, 0FF01h ; add free clusters
4407 0000B691 E863080000    <1>      call    calculate_fat_freespace
4408                      <1>
4409                      <1>      ;inc     eax ; 0FFFFFFFh -> 0 ; recalculation is needed!
4410                      <1>      ;jnz     short loc_createfile_save_fat_buffer_2
4411                      <1>
4412                      <1>      ; ecx > 0 -> Recalculation is needed
4413 0000B696 09C9      <1>      or      ecx, ecx
4414 0000B698 7409      <1>      jz      short loc_createfile_save_fat_buffer_2
4415                      <1>
4416 0000B69A 66BB00FF      <1>      mov     bx, 0FF00h ; ; recalculate free space
4417 0000B69E 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 0000B6A3 E82C040000    <1>      call    get_first_free_cluster
4424 0000B6A8 72C6      <1>      jc      short loc_createfile_gffc_2_stc_retn
4425                      <1>
4426 0000B6AA A3[38450100]    <1>      mov     [createfile_FFCluster], eax
4427                      <1>
4428 0000B6AF A1[3C450100]    <1>      mov     eax, [createfile_LastDirCluster]
4429                      <1>
4430 0000B6B4 E8AA030000    <1>      call    load_FAT_sub_directory
4431 0000B6B9 72B5      <1>      jc      short loc_createfile_gffc_2_stc_retn
4432                      <1>
4433 0000B6BB BF00000800    <1>      mov     edi, Directory_Buffer
4434                      <1>
4435 0000B6C0 6629DB      <1>      sub     bx, bx ; directory entry index/number = 0
4436                      <1>
4437 0000B6C3 56          <1>      push    esi ; * ; 23/03/2016
4438                      <1>
4439                      <1> loc_createfile_set_ff_dir_entry:
4440 0000B6C4 66891D[4A450100] <1>      mov     [createfile_DirIndex], bx
4441                      <1>
4442                      <1>      ; EDI = Directory entry address
4443 0000B6CB 8B35[2C450100]    <1>      mov     esi, [createfile_Name_Offset]
4444 0000B6D1 A1[38450100]    <1>      mov     eax, [createfile_FFCluster]
4445 0000B6D6 A3[40450100]    <1>      mov     [createfile_Cluster], eax ; 24/03/2016
4446 0000B6DB B5FF      <1>      mov     ch, 0FFh
4447 0000B6DD 8A0D[48450100]    <1>      mov     cl, [createfile_attr] ; file attributes
4448                      <1>      ; CH > 0 -> File size is in [EBX]
4449 0000B6E3 BB[34450100] <1>      mov     ebx, createfile_size
4450                      <1>
4451 0000B6E8 E800EEFFFF      <1>      call    make_directory_entry
4452                      <1>
4453 0000B6ED 5E          <1>      pop     esi ; * ; ESI = Logical Dos Drv Desc. Table address
4454                      <1>
4455 0000B6EE C605[10410100]02 <1>      mov     byte [DirBuff_ValidData], 2
4456 0000B6F5 E8D0EEFFFF      <1>      call    save_directory_buffer
4457 0000B6FA 7221      <1>      jc      short loc_createfile_set_ff_dir_entry_retn
4458                      <1>
4459 0000B6FC C605[53450100]01 <1>      mov     byte [createfile_UpdatePDir], 1 ; 31/03/2016
4460                      <1>
4461                      <1> loc_createfile_get_set_write_file_cluster:
4462 0000B703 A1[34450100]    <1>      mov     eax, [createfile_size]
4463 0000B708 09C0      <1>      or      eax, eax
4464 0000B70A 7570      <1>      jnz     short loc_createfile_get_set_wfc_cont

```

```

4465 0000B70C 40          <1>      inc     eax
4466                      <1>      ; 23/03/2016
4467 0000B70D 0FB61D[49450100] <1>      movzx  ebx, byte [createfile_SecPerClust]
4468                      <1>      ;movzx ecx, word [esi+LD_BPB+BytesPerSec] ; 512
4469 0000B714 0FB70D[50450100] <1>      movzx  ecx, word [createfile_BytesPerSec] ; 512
4470 0000B71B EB7C          <1>      jmp     loc_createfile_set_cluster_count
4471                      <1>
4472                      <1> loc_createfile_set_ff_dir_entry_retn:
4473 0000B71D C3            <1>      retn
4474                      <1>
4475                      <1> loc_createfile_write_fcluster_to_disk:
4476 0000B71E 034668        <1>      add     eax, [esi+LD_DATABegin] ; convert to physical address
4477 0000B721 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 0000B726 E89D3A0000    <1>      call    disk_write
4483 0000B72B 7211          <1>      jc      short loc_createfile_dsk_wr_err
4484                      <1>
4485                      <1> loc_createfile_update_fat_cluster:
4486                      <1>      ; 21/03/2016
4487 0000B72D 803D[52450100]00 <1>      cmp     byte [createfile_wfc], 0
4488 0000B734 7712          <1>      ja      short loc_createfile_update_fat_cluster_n1
4489                      <1>
4490 0000B736 FE05[52450100] <1>      inc     byte [createfile_wfc] ; 1
4491 0000B73C 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 0000B73E B812000000    <1>      mov     eax, 18 ; Drive not ready or write error !
4497 0000B743 E9BD000000    <1>      jmp     loc_createfile_stc_retn
4498                      <1>
4499                      <1> loc_createfile_update_fat_cluster_n1:
4500 0000B748 A1[44450100]   <1>      mov     eax, [createfile_PCluster]
4501 0000B74D 8B0D[40450100] <1>      mov     ecx, [createfile_Cluster]
4502 0000B753 E84F040000    <1>      call    update_cluster
4503 0000B758 7308          <1>      jnc     short loc_createfile_update_fat_cluster_n2
4504 0000B75A 09C0          <1>      or      eax, eax ; EAX = 0 -> cluster value is 0 or eocc
4505 0000B75C 0F85A3000000  <1>      jnz     loc_createfile_stc_retn
4506                      <1>
4507                      <1> loc_createfile_update_fat_cluster_n2:
4508 0000B762 A1[40450100]   <1>      mov     eax, [createfile_Cluster]
4509 0000B767 B9FFFFFF0F      <1>      mov     ecx, 0FFFFFFFh
4510 0000B76C E836040000    <1>      call    update_cluster
4511 0000B771 734E          <1>      jnc     short loc_createfile_save_fat_buffer_3
4512 0000B773 09C0          <1>      or      eax, eax ; EAX = 0 -> cluster value is 0 or eocc
4513 0000B775 744A          <1>      jz      short loc_createfile_save_fat_buffer_3
4514                      <1>
4515                      <1> loc_createfile_upd_fat_fcluster_stc_retn:
4516 0000B777 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 0000B77C 0FB70D[50450100] <1>      movzx  ecx, word [createfile_BytesPerSec] ; 512
4521 0000B783 01C8          <1>      add     eax, ecx
4522 0000B785 48            <1>      dec     eax ; add eax, 511
4523 0000B786 29D2          <1>      sub     edx, edx
4524 0000B788 F7F1          <1>      div     ecx
4525 0000B78A 0FB61D[49450100] <1>      movzx  ebx, byte [createfile_SecPerClust]
4526 0000B791 01D8          <1>      add     eax, ebx
4527 0000B793 48            <1>      dec     eax ; add eax, SecPerClust - 1
4528 0000B794 6631D2        <1>      xor     dx, dx
4529 0000B797 F7F3          <1>      div     ebx
4530                      <1>
4531                      <1> loc_createfile_set_cluster_count:
4532 0000B799 A3[4C450100]   <1>      mov     [createfile_CCount], eax
4533                      <1>
4534 0000B79E BF00000700    <1>      mov     edi, Cluster_Buffer
4535 0000B7A3 89C8          <1>      mov     eax, ecx ; Bytes per Sector
4536 0000B7A5 F7E3          <1>      mul     ebx ; Sectors per Cluster
4537                      <1>      ; EAX = Bytes per Cluster
4538 0000B7A7 89C1          <1>      mov     ecx, eax
4539 0000B7A9 C1E902        <1>      shr     ecx, 2 ; dword count
4540 0000B7AC 31C0          <1>      xor     eax, eax
4541 0000B7AE F3AB          <1>      rep     stosd ; clear cluster buffer
4542                      <1>
4543 0000B7B0 A1[40450100]   <1>      mov     eax, [createfile_Cluster] ; 24/03/2016
4544                      <1>
4545 0000B7B5 89D9          <1>      mov     ecx, ebx
4546                      <1>
4547                      <1> loc_createfile_get_set_wf_fclust_cont:
4548 0000B7B7 83E802        <1>      sub     eax, 2
4549 0000B7BA F7E1          <1>      mul     ecx
4550                      <1>      ; EAX = Logical DOS disk address (offset)
4551 0000B7BC 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 0000B7C1 E89E060000    <1>      call    save_fat_buffer
4556 0000B7C6 723D          <1>      jc      loc_createfile_stc_retn
4557                      <1>
4558                      <1>      ; 21/03/2016
4559 0000B7C8 803D[06410100]01 <1>      cmp     byte [FAT_ClusterCounter], 1
4560 0000B7CF 721B          <1>      jb      short loc_createfile_save_fat_buffer_4
4561                      <1>
4562                      <1>      ; ESI = Logical DOS Drive Description Table address
4563 0000B7D1 A1[06410100]   <1>      mov     eax, [FAT_ClusterCounter]
4564 0000B7D6 66BB01FF      <1>      mov     bx, 0FF01h ; add free clusters
4565 0000B7DA E81A070000    <1>      call    calculate_fat_freespace
4566                      <1>

```



```

4567      <1>      ;inc  eax ; 0FFFFFFFh -> 0 ; recalculation is needed!
4568      <1>      ;jnz  short loc_createfile_save_fat_buffer_4
4569      <1>
4570      <1>      ; ecx > 0 -> Recalculation is needed
4571      <1>      or    ecx, ecx
4572      <1>      jz    short loc_createfile_save_fat_buffer_4
4573      <1>
4574      <1>      mov   bx, 0FF00h ; ; recalculate free space
4575      <1>      call  calculate_fat_freespace
4576      <1>
4577      <1> loc_createfile_save_fat_buffer_4:
4578      <1>      dec   dword [createfile_CCount]
4579      <1>      ;jz    short loc_createfile_upd_dir_modif_date_time
4580      <1>      jz    short loc_createfile_stc_retn_cc ; 31/03/2016
4581      <1>
4582      <1> loc_createfile_get_set_write_next_cluster:
4583      <1>      call  get_first_free_cluster
4584      <1>      jc    short loc_createfile_stc_retn
4585      <1>
4586      <1> loc_createfile_get_set_write_next_cluster_1:
4587      <1>      cmp   eax, 0FFFFFFFh
4588      <1>      jb    short loc_createfile_get_set_write_next_cluster_2
4589      <1>
4590      <1> loc_createfile_wnc_insufficient_disk_space:
4591      <1>      mov   eax, 27h ; Insufficient disk space
4592      <1>
4593      <1> loc_createfile_stc_retn:
4594      <1>      cmp   byte [createfile_wfc], 1
4595      <1>      jnb   short loc_createfile_err_retn
4596      <1>      retn
4597      <1>
4598      <1> loc_createfile_wnc_inv_format_retn:
4599      <1>      ;mov   eax, 28
4600      <1>      mov   al, 28 ; Invalid format
4601      <1>      jmp   short loc_createfile_stc_retn
4602      <1>
4603      <1> loc_createfile_get_set_write_next_cluster_2:
4604      <1>      cmp   eax, 2
4605      <1>      jb    short loc_createfile_wnc_inv_format_retn
4606      <1>
4607      <1> loc_createfile_get_set_write_next_cluster_3:
4608      <1>      mov   ecx, [createfile_Cluster]
4609      <1>      mov   [createfile_Cluster], eax
4610      <1>      mov   [createfile_PCluster], ecx
4611      <1>      movzx ecx, byte [createfile_SecPerClust]
4612      <1>      jmp   short loc_createfile_get_set_wf_fclust_cont
4613      <1>
4614      <1> loc_createfile_err_retn:
4615      <1>      stc
4616      <1>
4617      <1> ;loc_createfile_upd_dir_modif_date_time:
4618      <1> loc_createfile_stc_retn_cc: ; 31/03/2016
4619      <1>      pushf ; cpu is here for an error return or completion
4620      <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      <1>      mov   eax, [FAT_ClusterCounter]
4626      <1>      or    eax, eax
4627      <1>      jz    short loc_createfile_stc_retn_pop_eax
4628      <1>      mov   bh, [Current_Drv]
4629      <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      <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      <1>      and   ecx, ecx ; cx = 0 -> valid free sector count
4636      <1>      jz    short loc_createfile_stc_retn_pop_eax
4637      <1>
4638      <1> loc_createfile_stc_retn_recalc_FAT_freespace:
4639      <1>      mov   bx, 0FF00h ; bh = 0FFh ->
4640      <1>      ; ESI = Logical DOS Drv DT Addr
4641      <1>      ; BL = 0 -> Recalculate
4642      <1>      call  calculate_fat_freespace
4643      <1>
4644      <1> loc_createfile_stc_retn_pop_eax:
4645      <1>      pop   eax
4646      <1>      popf
4647      <1>      jc    short loc_createfile_retn
4648      <1>
4649      <1> loc_createfile_retn_fcluster:
4650      <1>      mov   eax, [createfile_FFCluster]
4651      <1>      mov   ebx, createfile_size
4652      <1>      ;movzx ecx, byte [esi+LD_BPB+SecPerClust]
4653      <1>      movzx ecx, byte [createfile_SecPerClust] ; 23/03/2016
4654      <1>      movzx edx, word [createfile_DirIndex]
4655      <1>
4656      <1> loc_createfile_retn:
4657      <1>      retn
4658      <1>
4659      <1> create_fs_file:
4660      <1>      ; temporary (21/03/2016)
4661      <1>      retn
4662      <1>
4663      <1> delete_fs_file:
4664      <1>      ; temporary (28/02/2016)
4665      <1>      retn
4666      <1>
4667      <1> rename_fs_file_or_directory:
4668      <1>      retn

```

```

4669      <1>
4670      <1> make_fs_directory:
4671      <1>         ; temporary (21/02/2016)
4672 0000B878 C3      <1>         retn
4673      <1>
4674      <1> add_new_fs_section:
4675      <1>         ; temporary (11/03/2016)
4676 0000B879 C3      <1>         retn
4677      <1>
4678      <1> delete_fs_directory_entry:
4679      <1>         ; temporary (11/03/2016)
4680 0000B87A C3      <1>         retn
4681      <1>
4682      <1> csftdf2_read_fs_file_sectors:
4683      <1>         ; temporary (19/03/2016)
4684 0000B87B C3      <1>         retn
4685      <1>
4686      <1> csftdf2_write_fs_file_sectors:
4687      <1>         ; temporary (19/03/2016)
4688 0000B87C C3      <1>         retn
2308      <1> %include 'trdosk5.s' ; 24/01/2016
      <1> ; *****
      <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - File System Procedures : trdosk5s
      <1> ; -----
      <1> ; Last Update: 23/10/2016
      <1> ; -----
      <1> ; Beginning: 24/01/2016
      <1> ; -----
      <1> ; Assembler: NASM version 2.11 (trdos386.s)
      <1> ; -----
      <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
      <1> ; DRV_FAT.ASM (21/08/2011)
      <1> ; *****
      <1> ; DRV_FAT.ASM (c) 2005-2011 Erdogan TAN [ 07/07/2009 ] Last Update: 21/08/2011
      <1>
      <1> get_next_cluster:
      <1>         ; 15/10/2016
      <1>         ; 23/03/2016
      <1>         ; 01/02/2016 (TRDOS 386 = TRDOS v2.0)
      <1>         ; 05/07/2011
      <1>         ; 07/07/2009
      <1>         ; 2005
      <1>         ; INPUT ->
      <1>         ;     EAX = Cluster Number (32 bit)
      <1>         ;     ESI = Logical DOS Drive Parameters Table
      <1>         ; OUTPUT ->
      <1>         ;     cf = 0 -> No Error, EAX valid
      <1>         ;     cf = 1 & EAX = 0 -> End Of Cluster Chain
      <1>         ;     cf = 1 & EAX > 0 -> Error
      <1>         ;     ECX = Current/Previous cluster (if CF = 0)
      <1>         ;     EAX = Next Cluster Number (32 bit)
      <1>         ;
      <1>         ; (Modified registers: EAX, ECX, EBX, EDX)
      <1>
      <1>         mov     [FAT_CurrentCluster], eax
      <1> check_next_cluster_fat_type:
      <1>         sub     edx, edx ; 0
      <1>         cmp     byte [esi+LD_FATType], 2
      <1>         jnb    short get_FAT12_next_cluster
      <1>         ja     get_FAT32_next_cluster
      <1> get_FAT16_next_cluster:
      <1>         mov     ebx, 300h ;768
      <1>         div     ebx
      <1>         ; EAX = Count of 3 FAT sectors
      <1>         ; EDX = Cluster Offset (< 768)
      <1>         shl     dx, 1 ; Multiply by 2
      <1>         mov     ebx, edx ; Byte Offset
      <1>         add     ebx, FAT_Buffer
      <1>         mov     dx, 3
      <1>         mul     edx
      <1>         ; EAX = FAT Sector (<= 256)
      <1>         ; EDX = 0
      <1>         mov     cl, [esi+LD_Name]
      <1>         cmp     byte [FAT_BuffValidData], 0
      <1>         jna     load_FAT_sectors0
      <1>         cmp     cl, [FAT_BuffDrvName]
      <1>         jne     load_FAT_sectors0
      <1>         cmp     eax, [FAT_BuffSector]
      <1>         jne     load_FAT_sectors1
      <1>         ;movzx eax, word [ebx]
      <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
      <1>         cmp     ax, 0FFF7h
      <1>         jb     short loc_pass_gnc_FAT16_eoc_check
      <1>         ; ax >= FFF7h (cluster 0002h to FFF6h is valid, in use)
      <1>         jmp     short loc_pass_gnc_FAT16_eoc_check_xor_eax
      <1>
      <1> get_FAT12_next_cluster:
      <1>         mov     ebx, 400h ;1024
      <1>         div     ebx
      <1>         ; EAX = Count of 3 FAT sectors
      <1>         ; EDX = Cluster Offset (< 1024)
      <1>         push    ax
      <1>         mov     ax, 3
      <1>         mul     dx ; Multiply by 3

```

```

82 0000B8EA 66D1E8      <1>      shr     ax, 1 ; Divide by 2
83 0000B8ED 6689C3      <1>      mov     bx, ax ; Byte Offset
84 0000B8F0 81C3001C0900  <1>      add     ebx, FAT_Buffer
85 0000B8F6 6658          <1>      pop     ax
86 0000B8F8 66BA0300      <1>      mov     dx, 3
87 0000B8FC F7E2          <1>      mul     edx
88              <1>      ; EAX = FAT Sector (<= 12)
89              <1>      ; EDX = 0
90 0000B8FE 8A0E          <1>      mov     cl, [esi+LD_Name]
91 0000B900 803D[FE400100]00 <1>      cmp     byte [FAT_BuffValidData], 0
92 0000B907 767A          <1>      jna     short load_FAT_sectors0
93 0000B909 3A0D[FF400100] <1>      cmp     cl, [FAT_BuffDrvName]
94 0000B90F 7572          <1>      jne     short load_FAT_sectors0
95 0000B911 3B05[02410100] <1>      cmp     eax, [FAT_BuffSector]
96 0000B917 7570          <1>      jne     short load_FAT_sectors1
97 0000B919 A1[FA400100] <1>      mov     eax, [FAT_CurrentCluster]
98 0000B91E 66D1E8      <1>      shr     ax, 1
99              <1>      ;movzx eax, word [ebx]
100 0000B921 668B03      <1>      mov     ax, [ebx]
101 0000B924 7314          <1>      jnc     short get_FAT12_nc_even
102 0000B926 66C1E804      <1>      shr     ax, 4
103              <1> loc_gnc_fat12_eoc_check:
104              <1>      ;cmp     al, 0F7h
105              <1>      ;jb     short loc_pass_gnc_FAT16_eoc_check
106              <1>      ;cmp     ah, 0Fh
107              <1>      ;jb     short loc_pass_gnc_FAT16_eoc_check
108 0000B92A 663DF70F      <1>      cmp     ax, 0FF7h
109 0000B92E 7202          <1>      jb     short loc_pass_gnc_FAT16_eoc_check
110              <1>      ; ax >= FF7h (cluster 0002h to FF6h is valid, in use)
111              <1>
112              <1> loc_pass_gnc_FAT16_eoc_check_xor_eax:
113 0000B930 31C0          <1>      xor     eax, eax ; 0
114              <1> loc_pass_gnc_FAT16_eoc_check:
115              <1> loc_pass_gnc_FAT32_eoc_check:
116 0000B932 8B0D[FA400100] <1>      mov     ecx, [FAT_CurrentCluster]
117 0000B938 F5          <1>      cmc
118 0000B939 C3          <1>      retn
119              <1>
120              <1> get_FAT12_nc_even:
121 0000B93A 80E40F      <1>      and     ah, 0Fh
122 0000B93D EBEB          <1>      jmp     short loc_gnc_fat12_eoc_check
123              <1>
124              <1> get_FAT32_next_cluster:
125 0000B93F BB80010000 <1>      mov     ebx, 180h ;384
126 0000B944 F7F3          <1>      div     ebx
127              <1>      ; EAX = Count of 3 FAT sectors
128              <1>      ; EDX = Cluster Offset (< 384)
129 0000B946 66C1E202      <1>      shl     dx, 2 ; Multiply by 4
130 0000B94A 89D3          <1>      mov     ebx, edx ; Byte Offset
131 0000B94C 81C3001C0900 <1>      add     ebx, FAT_Buffer
132 0000B952 66BA0300      <1>      mov     dx, 3
133 0000B956 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 0000B958 8A0E          <1>      mov     cl, [esi+LD_Name]
139 0000B95A 803D[FE400100]00 <1>      cmp     byte [FAT_BuffValidData], 0
140 0000B961 7620          <1>      jna     short load_FAT_sectors0
141 0000B963 3A0D[FF400100] <1>      cmp     cl, [FAT_BuffDrvName]
142 0000B969 7518          <1>      jne     short load_FAT_sectors0
143 0000B96B 3B05[02410100] <1>      cmp     eax, [FAT_BuffSector] ; 0, 3, 6, 9 ...
144 0000B971 7516          <1>      jne     short load_FAT_sectors1
145 0000B973 8B03          <1>      mov     eax, [ebx]
146 0000B975 25FFFFFFF0F <1>      and     eax, 0FFFFFFFh ; 28 bit Cluster
147 0000B97A 3DF7FFFF0F <1>      cmp     eax, 0FFFFFFFh
148 0000B97F 72B1          <1>      jb     short loc_pass_gnc_FAT32_eoc_check
149              <1>      ; eax >= FFFFFFF7h (cluster 0002h to FFFFFFF6h is valid)
150 0000B981 EBAD          <1>      jmp     short loc_pass_gnc_FAT16_eoc_check_xor_eax
151              <1>
152              <1> load_FAT_sectors0:
153 0000B983 880D[FF400100] <1>      mov     [FAT_BuffDrvName], cl
154              <1> load_FAT_sectors1:
155 0000B989 A3[02410100] <1>      mov     [FAT_BuffSector], eax
156 0000B98E 89C3          <1>      mov     ebx, eax
157 0000B990 034660      <1>      add     eax, [esi+LD_FATBegin]
158 0000B993 807E0302      <1>      cmp     byte [esi+LD_FATType], 2
159 0000B997 7706          <1>      ja     short load_FAT_sectors3
160 0000B999 0FB74E1C      <1>      movzx   ecx, word [esi+LD_BPB+BPB_FATSz16]
161 0000B99D EB03          <1>      jmp     short load_FAT_sectors4
162              <1> load_FAT_sectors3:
163 0000B99F 8B4E2A      <1>      mov     ecx, [esi+LD_BPB+BPB_FATSz32]
164              <1> load_FAT_sectors4:
165 0000B9A2 29D9          <1>      sub     ecx, ebx ; [FAT_BuffSector]
166 0000B9A4 83F903      <1>      cmp     ecx, 3
167 0000B9A7 7605          <1>      jna     short load_FAT_sectors5
168 0000B9A9 B903000000      <1>      mov     ecx, 3
169              <1> load_FAT_sectors5:
170 0000B9AE BB001C0900 <1>      mov     ebx, FAT_Buffer
171 0000B9B3 E81F380000 <1>      call    disk_read
172 0000B9B8 730D          <1>      jnc     short load_FAT_sectors_ok
173              <1>      ; 15/10/2016 (15h -> 17)
174              <1>      ; 23/03/2016 (15h)
175 0000B9BA B811000000 <1>      mov     eax, 17 ; Drive not ready or read error
176 0000B9BF C605[FE400100]00 <1>      mov     byte [FAT_BuffValidData], 0
177 0000B9C6 C3          <1>      retn
178              <1> load_FAT_sectors_ok:
179 0000B9C7 C605[FE400100]01 <1>      mov     byte [FAT_BuffValidData], 1
180 0000B9CE A1[FA400100] <1>      mov     eax, [FAT_CurrentCluster]
181 0000B9D3 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      0000B9D8 8A1E      <1>      mov     bl, [esi+LD_Name]
208      0000B9DA 8A7E03    <1>      mov     bh, [esi+LD_FATType]
209      <1>
210      <1>      ;mov     [DirBuff_DRV], bl
211      <1>      ;mov     [DirBuff_FATType], bh
212      0000B9DD 66891D[0E410100] <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      0000B9E4 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      0000B9E8 6681FA0002 <1>      cmp     dx, 512 ; Number of Root Dir Entries
224      0000B9ED 7414      <1>      je      short lrd_mov_ecx_32
225      0000B9EF 89D0      <1>      mov     eax, edx
226      <1>      ; 23/10/2016
227      0000B9F1 89C1      <1>      mov     ecx, eax
228      0000B9F3 6683C10F    <1>      add     cx, 15 ; round up
229      0000B9F7 66C1E904    <1>      shr     cx, 4 ; 16 entries per sector (512/32)
230      <1>      ; ecx = Root directory size in sectors
231      0000B9FB 66C1E005    <1>      shl     ax, 5 ; Root directory size in bytes
232      0000B9FF 664A      <1>      dec     dx ; Last entry number of root dir
233      <1>      ; cx = Dir Buffer sector count
234      0000BA01 EB0B      <1>      jmp     short lrd_check_dir_buffer
235      <1>
236      <1> lrd_mov_ecx_32:
237      0000BA03 B920000000    <1>      mov     ecx, 32
238      0000BA08 664A      <1>      dec     dx ; 511
239      0000BA0A 66B80040    <1>      mov     ax, 32*512
240      <1>
241      <1> lrd_check_dir_buffer:
242      0000BA0E 29DB      <1>      sub     ebx, ebx ; 0
243      0000BA10 881D[10410100] <1>      mov     [DirBuff_ValidData], bl ; 0
244      0000BA16 668915[13410100] <1>      mov     [DirBuff_LastEntry], dx
245      0000BA1D 891D[15410100] <1>      mov     [DirBuff_Cluster], ebx ; 0
246      0000BA23 66A3[19410100] <1>      mov     [DirBuffer_Size], ax
247      <1>
248      0000BA29 8B4664    <1>      mov     eax, [esi+LD_ROOTBegin]
249      <1> read_directory:
250      0000BA2C BB00000800    <1>      mov     ebx, Directory_Buffer
251      0000BA31 51      <1>      push    ecx ; Directory buffer sector count
252      0000BA32 53      <1>      push    ebx
253      0000BA33 E89F370000    <1>      call    disk_read
254      0000BA38 5B      <1>      pop     ebx
255      0000BA39 720B      <1>      jc      short load_DirBuff_error
256      <1>
257      <1> validate_DirBuff_and_return:
258      0000BA3B 59      <1>      pop     ecx ; Number of loaded sectors
259      0000BA3C C605[10410100]01 <1>      mov     byte [DirBuff_ValidData], 1
260      0000BA43 31C0      <1>      xor     eax, eax ; 0 = no error
261      0000BA45 C3      <1>      retn
262      <1>
263      <1> load_DirBuff_error:
264      0000BA46 89C8      <1>      mov     eax, ecx ; remaining sectors
265      0000BA48 59      <1>      pop     ecx ; sector count
266      0000BA49 29C1      <1>      sub     ecx, eax ; Number of loaded sectors
267      <1>      ; 15/10/2016 (15h -> 17)
268      0000BA4B B811000000    <1>      mov     eax, 17 ; DRV NOT READY OR READ ERROR !
269      0000BA50 F9      <1>      stc
270      0000BA51 C3      <1>      retn
271      <1>
272      <1> load_FAT32_root_directory:
273      <1>      ; 02/02/2016 (TRDOS 386 = TRDOS v2.0)
274      <1>      ;
275      <1>      ; INPUT ->
276      <1>      ;     ESI = Logical DOS Drive Description Table
277      <1>      ; OUTPUT ->
278      <1>      ;     cf = 1 -> Root directory could not be loaded
279      <1>      ;     EAX > 0 -> Error number
280      <1>      ;     cf = 0 -> EAX = 0
281      <1>      ;     ECX = Directory buffer size in sectors (CL)
282      <1>      ;     EBX = Directory buffer address
283      <1>      ;     NOTE: DirBuffer_Size is in bytes ! (word)
284      <1>      ;
285      <1>      ; (Modified registers: EAX, ECX, EBX, EDX)

```



```

286 <1>
287 <1>
288 0000BA52 8A1E <1> mov bl, [esi+LD_Name]
289 0000BA54 8A7E03 <1> mov bh, [esi+LD_FATType]
290 <1>
291 <1> ;mov [DirBuff_DRV], bl
292 <1> ;mov [DirBuff_FATType], bh
293 0000BA57 66891D[0E410100] <1> mov [DirBuff_DRV], bx
294 <1>
295 <1> load_FAT32_root_dir0:
296 0000BA5E 8B4632 <1> mov eax, [esi+LD_BPB+FAT32_RootFClust]
297 0000BA61 EB0C <1> jmp short load_FAT_sub_dir0
298 <1>
299 <1> load_FAT_sub_directory:
300 <1> ; 01/02/2016 (TRDOS 386 = TRDOS v2.0)
301 <1> ; 05/07/2011
302 <1> ; 23/08/2009
303 <1> ;
304 <1> ; INPUT ->
305 <1> ; ESI = Logical DOS Drive Description Table
306 <1> ; EAX = Cluster Number
307 <1> ; OUTPUT ->
308 <1> ; cf = 1 -> Sub directory could not be loaded
309 <1> ; EAX > 0 -> Error number
310 <1> ; cf = 0 -> EAX = 0
311 <1> ; ECX = Directory buffer size in sectors (CL)
312 <1> ; EBX = Directory buffer address
313 <1> ;
314 <1> ; NOTE: DirBuffer_Size is in bytes ! (word)
315 <1> ;
316 <1> ; (Modified registers: EAX, ECX, EBX, EDX)
317 <1>
318 0000BA63 8A1E <1> mov bl, [esi+LD_Name]
319 0000BA65 8A7E03 <1> mov bh, [esi+LD_FATType]
320 <1>
321 <1> ;mov [DirBuff_DRV], bl
322 <1> ;mov [DirBuff_FATType], bh
323 0000BA68 66891D[0E410100] <1> mov [DirBuff_DRV], bx
324 <1>
325 <1> load_FAT_sub_dir0:
326 0000BA6F 0FB64E13 <1> movzx ecx, byte [esi+LD_BPB+SecPerClust]
327 <1>
328 0000BA73 882D[10410100] <1> mov [DirBuff_ValidData], ch ; 0
329 0000BA79 A3[15410100] <1> mov [DirBuff_Cluster], eax
330 <1>
331 0000BA7E 0FB74611 <1> movzx eax, word [esi+LD_BPB+BytesPerSec]
332 0000BA82 F7E1 <1> mul ecx
333 0000BA84 C1E805 <1> shr eax, 5 ; directory entry count (dir size / 32)
334 0000BA87 6648 <1> dec ax ; last entry
335 0000BA89 66A3[13410100] <1> mov [DirBuff_LastEntry], ax
336 <1>
337 0000BA8F A1[15410100] <1> mov eax, [DirBuff_Cluster]
338 0000BA94 83E802 <1> sub eax, 2
339 0000BA97 F7E1 <1> mul ecx
340 0000BA99 034668 <1> add eax, [esi+LD_DATABegin]
341 <1> ; ecx = sector per cluster (dir buffer size = 32 sectors)
342 0000BA9C EB8E <1> jmp short read_directory
343 <1>
344 <1> ; DRV_FS.ASM
345 <1>
346 <1> load_current_FS_directory:
347 0000BA9E C3 <1> retn
348 <1> load_FS_root_directory:
349 0000BA9F C3 <1> retn
350 <1> load_FS_sub_directory:
351 0000BAA0 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 0000BAA1 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 0000BAA5 807E0300 <1> cmp byte [esi+LD_FATType], 0
378 0000BAA9 761C <1> jna short read_fs_cluster
379 <1>
380 <1> read_fat_file_sectors: ; 18/03/2016
381 0000BAAB 83E802 <1> sub eax, 2 ; Beginning cluster number is always 2
382 0000BAAE 0FB65613 <1> movzx edx, byte [esi+LD_BPB+BPB_SecPerClust] ; 18/03/2016
383 0000BAB2 F7E2 <1> mul edx
384 0000BAB4 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      0000BAB7 E81B370000      <1>      call    disk_read
393      0000BABC 7306           <1>      jnc     short rclust_retn
394      <1>
395      <1>      ; 15/10/2016 (15h -> 17)
396      0000BABE B811000000      <1>      mov     eax, 17 ; Drive not ready or read error !
397      0000BAC3 C3             <1>      retn
398      <1>
399      <1> rclust_retn:
400      0000BAC4 29C0           <1>      sub     eax, eax ; 0
401      0000BAC6 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      0000BAC7 B980000000      <1>      mov     ecx, 128 ; maximum count of sectors (before eof)
416      0000BACC E801000000      <1>      call    read_fs_sectors
417      0000BAD1 C3             <1>      retn
418      <1>
419      <1> read_fs_sectors:
420      <1>      ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
421      0000BAD2 F9             <1>      stc
422      0000BAD3 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
436      <1>      ; If the drive has FAT32 fs:
437      <1>      ; EBX = FAT32 FSI sector buffer address (if > 0)
438      <1>
439      0000BAD4 8B4678          <1>      mov     eax, [esi+LD_Clusters]
440      0000BAD7 40              <1>      inc     eax ; add eax, 1
441      0000BAD8 A3[98430100]    <1>      mov     [gffc_last_free_cluster], eax
442      <1>
443      0000BADD 31DB            <1>      xor     ebx, ebx ; 0 ; 02/03/2016
444      <1>
445      0000BADF 807E0302        <1>      cmp     byte [esi+LD_FATType], 2
446      0000BAE3 760E            <1>      jna     short loc_gffc_get_first_fat_free_cluster0
447      <1>
448      <1> loc_gffc_get_first_fat32_free_cluster:
449      <1>      ; 02/03/2016
450      0000BAE5 E844060000      <1>      call    get_fat32_fsinfo_sector_parms
451      0000BAEA 7207            <1>      jc      short loc_gffc_get_first_fat_free_cluster0
452      <1>
453      <1> loc_gffc_check_fsinfo_parms:
454      <1>      ;mov ebx, DOSBootSectorBuff
455      <1>      ;cmp dword [ebx], 41615252h
456      <1>      ;jne short loc_gffc_fat32_fsinfo_err
457      <1>      ;cmp dword [ebx+484], 61417272h
458      <1>      ;jne short loc_gffc_fat32_fsinfo_err
459      <1>      ;mov eax, [ebx+492] ; FSI_Next_Free
460      <1>      ;EAX = First free cluster
461      <1>      ;(from FAT32 FSInfo sector)
462      0000BAEC 89D0            <1>      mov     eax, edx ; FSI_Next_Free (First Free Cluster)
463      0000BAEE 83F8FF          <1>      cmp     eax, 0FFFFFFFh ; invalid (unknown) !
464      0000BAF1 7205            <1>      jb      short loc_gffc_get_first_fat_free_cluster1
465      <1>
466      <1>      ; Start from the 1st cluster of the FAT(32) file system
467      <1> loc_gffc_get_first_fat_free_cluster0:
468      0000BAF3 B802000000      <1>      mov     eax, 2
469      <1>      ;xor edx, edx
470      <1>
471      <1> loc_gffc_get_first_fat_free_cluster1:
472      0000BAF8 53             <1>      push    ebx ; 02/03/2016
473      <1>
474      <1> loc_gffc_get_first_fat_free_cluster2:
475      0000BAF9 A3[94430100]    <1>      mov     [gffc_first_free_cluster], eax
476      0000BAFE A3[90430100]    <1>      mov     [gffc_next_free_cluster], eax
477      <1>
478      <1>      ; EBX = FAT32 FSINFO sector buffer address
479      <1>      ; (EBX = 0, if the drive has not got FAT32 fs or
480      <1>      ; FAT32 FSINFO sector buffer is invalid.)
481      <1>
482      <1> loc_gffc_get_first_fat_free_cluster3:
483      0000BB03 E875FDFFFF      <1>      call    get_next_cluster
484      0000BB08 7307            <1>      jnc     short loc_gffc_get_first_fat_free_cluster4
485      0000BB0A 09C0            <1>      or      eax, eax
486      0000BB0C 740B            <1>      jz      short loc_gffc_first_free_fat_cluster_next
487      0000BB0E 5B             <1>      pop     ebx ; 02/03/2016
488      0000BB0F F5             <1>      cmc
489      0000BB10 C3             <1>      retn

```

334

490		<1>
491		<1> loc_gffc_get_first_fat_free_cluster4:
492	0000BB11 21C0	<1> and eax, eax ; next cluster value
493	0000BB13 7504	<1> jnz short loc_gffc_first_free_fat_cluster_next
494	0000BB15 89C8	<1> mov eax, ecx ; current (previous cluster) value
495	0000BB17 EB22	<1> jmp short loc_gffc_check_for_set
496		<1>
497		<1> loc_gffc_first_free_fat_cluster_next:
498	0000BB19 A1[90430100]	<1> mov eax, [gffc_next_free_cluster]
499	0000BB1E 3B05[98430100]	<1> cmp eax, [gffc_last_free_cluster]
500	0000BB24 7308	<1> jnb short retn_stc_from_get_first_free_cluster
501		<1> pass_gffc_last_cluster_eax_check:
502	0000BB26 40	<1> inc eax ; add eax, 1
503	0000BB27 A3[90430100]	<1> mov [gffc_next_free_cluster], eax
504	0000BB2C EBD5	<1> jmp short loc_gffc_get_first_fat_free_cluster3
505		<1>
506		<1> retn_stc_from_get_first_free_cluster:
507	0000BB2E A1[94430100]	<1> mov eax, [gffc_first_free_cluster]
508	0000BB33 83F802	<1> cmp eax, 2
509	0000BB36 7709	<1> ja short loc_gffc_check_previous_clusters
510	0000BB38 29C0	<1> sub eax, eax
511	0000BB3A 48	<1> dec eax ; FFFFFFFFh
512		<1>
513		<1> loc_gffc_check_for_set:
514		<1> ; 02/03/2016
515	0000BB3B 5B	<1> pop ebx
516		<1>
517		<1> ; EBX = FAT32 FSINFO sector buffer address
518		<1> ; (EBX = 0, if the drive has not got FAT32 fs or
519		<1> ; FAT32 FSINFO sector buffer is invalid.)
520		<1>
521	0000BB3C 09DB	<1> or ebx, ebx
522	0000BB3E 750E	<1> jnz short loc_gffc_set_ffree_fat32_cluster
523		<1>
524		<1> ;cmp byte [esi+LD_FATType], 3
525		<1> ;jnb short loc_gffc_set_ffree_fat32_cluster
526		<1>
527		<1> ;xor ebx, ebx ; 0
528		<1>
529		<1> loc_gffc_retn:
530	0000BB40 C3	<1> retn
531		<1>
532		<1> loc_gffc_check_previous_clusters:
533	0000BB41 48	<1> dec eax ; sub eax, 1
534	0000BB42 A3[98430100]	<1> mov [gffc_last_free_cluster], eax
535	0000BB47 B802000000	<1> mov eax, 2
536		<1> ;xor edx, edx
537	0000BB4C EBAB	<1> jmp short loc_gffc_get_first_fat_free_cluster2
538		<1>
539		<1> loc_gffc_set_ffree_fat32_cluster:
540		<1> ;call set_first_free_cluster
541		<1> ;retn
542		<1> ;jmp short set_first_free_cluster
543		<1>
544		<1> set_first_free_cluster:
545		<1> ; 15/10/2016
546		<1> ; 23/03/2016
547		<1> ; 02/03/2016
548		<1> ; 29/02/2016
549		<1> ; 26/02/2016
550		<1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
551		<1> ; 21/08/2011 (DRV_FAT.ASM, 'proc_set_first_free_cluster')
552		<1> ; 11/07/2010
553		<1> ; INPUT ->
554		<1> ; ESI = Logical DOS Drive Description Table address
555		<1> ; EAX = First free cluster
556		<1> ; EBX = FSINFO sector buffer address
557		<1> ; ;If EBX > 0, it is FSINFO sector buffer address
558		<1> ; ;EBX = 0, if FSINFO sector is not loaded
559		<1> ; OUTPUT->
560		<1> ; ESI = Logical DOS Drive Description Table address
561		<1> ; If EBX > 0, it is FSINFO sector buffer address
562		<1> ; EBX = 0, if FSINFO sector could not be loaded
563		<1> ; CF = 1 -> Error code in AL (EAX)
564		<1> ; CF = 0 -> first free cluster is successfully updated
565		<1>
566		<1> ;cmp byte [esi+LD_FATType], 3
567		<1> ;jb short loc_sffc_invalid_drive
568		<1>
569		<1> ; Save First Free Cluster value for 'update_cluster'
570	0000BB4E 89463E	<1> mov [esi+LD_BPB+BPB_Reserved+4], eax ; First free Cluster
571		<1>
572		<1> ;or ebx, ebx
573		<1> ;jnz short loc_sffc_read_fsinfo_sector
574		<1>
575	0000BB51 813B52526141	<1> cmp dword [ebx], 41615252h
576	0000BB57 7540	<1> jne short loc_sffc_read_fsinfo_sector
577	0000BB59 81BBE4010000727241-	<1> cmp dword [ebx+484], 61417272h
577	0000BB62 61	<1>
578	0000BB63 7534	<1> jne short loc_sffc_read_fsinfo_sector
579		<1>
580	0000BB65 3B83EC010000	<1> cmp eax, [ebx+492] ; FSI_Next_Free
581	0000BB6B 741F	<1> je short loc_sffc_retn
582		<1>
583		<1> loc_sffc_write_fsinfo_sector:
584		<1> ; EBX = FSINFO sector buffer
585		<1> ; [CFS_FAT32FSINFOSEC] is set in 'get_fat32_fsinfo_sector_parms'
586	0000BB6D 8983EC010000	<1> mov [ebx+492], eax
587	0000BB73 A1[A8430100]	<1> mov eax, [CFS_FAT32FSINFOSEC]
588	0000BB78 B901000000	<1> mov ecx, 1
589	0000BB7D 53	<1> push ebx
590	0000BB7E E845360000	<1> call disk_write

```

591 0000BB83 7208      <1>      jc      short loc_sffc_read_fsinfo_sector_err1
592 0000BB85 5B        <1>      pop     ebx
593                  <1>
594 0000BB86 8B83EC010000 <1>      mov     eax, [ebx+492] ; First (Next) Free Cluster
595                  <1>
596                  <1> loc_sffc_retn:
597 0000BB8C C3        <1>      retn
598                  <1>
599                  <1> ;loc_sffc_invalid_drive:
600                  <1> ;      mov     eax, 0Fh ; MSDOS Error : Invalid drive
601                  <1> ;      push    edx
602                  <1>
603                  <1> loc_sffc_read_fsinfo_sector_err1:
604 0000BB8D BB00000000 <1>      mov     ebx, 0
605                  <1>      ; 15/10/2016 (1Dh -> 18)
606                  <1>      ; 23/03/2016 (1Dh)
607 0000BB92 B812000000 <1>      mov     eax, 18 ; Drive not ready or write error
608                  <1>
609                  <1> loc_sffc_read_fsinfo_sector_err2:
610 0000BB97 5A        <1>      pop     edx
611 0000BB98 C3        <1>      retn
612                  <1>
613                  <1> loc_sffc_read_fsinfo_sector:
614 0000BB99 50        <1>      push    eax
615                  <1>
616 0000BB9A E88F050000 <1>      call    get_fat32_fsinfo_sector_parms
617 0000BB9F 72F6      <1>      jc      short loc_sffc_read_fsinfo_sector_err2
618                  <1>
619 0000BBA1 58        <1>      pop     eax
620                  <1>      ; EDX = First (Next) Free Cluster value from FSINFO sector
621                  <1>      ; EAX = First Free Cluster value from 'get_next_cluster'
622                  <1>      ; (edx = old value)
623 0000BBA2 39D0      <1>      cmp     eax, edx ; First free Cluster (eax = new value)
624 0000BBA4 75C7      <1>      jne     short loc_sffc_write_fsinfo_sector
625                  <1>
626 0000BBA6 C3        <1>      retn
627                  <1>
628                  <1> update_cluster:
629                  <1>      ; 23/10/2016
630                  <1>      ; 23/03/2016
631                  <1>      ; 02/03/2016
632                  <1>      ; 01/03/2016
633                  <1>      ; 29/02/2016
634                  <1>      ; 27/02/2016
635                  <1>      ; 26/02/2016
636                  <1>      ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
637                  <1>      ; 11/08/2011
638                  <1>      ; 09/02/2005
639                  <1>      ; INPUT ->
640                  <1>      ;      EAX = Cluster Number
641                  <1>      ;      ECX = New Cluster Value
642                  <1>      ;      ESI = Logical Dos Drive Parameters Table
643                  <1>      ;
644                  <1>      ;      /// dword [FAT_ClusterCounter] ///
645                  <1>      ;
646                  <1>      ; OUTPUT ->
647                  <1>      ;      cf = 0 -> No Error, EAX is valid
648                  <1>      ;      cf = 1 & EAX = 0 -> End Of Cluster Chain
649                  <1>      ;      cf = 1 & EAX > 0 -> Error
650                  <1>      ;      (ECX -> any value)
651                  <1>      ;      EAX = Next Cluster
652                  <1>      ;      ECX = New Cluster Value
653                  <1>      ;
654                  <1>      ;      /// [FAT_ClusterCounter] is updated,
655                  <1>      ;      /// decreased when a free cluster is assigned,
656                  <1>      ;      /// increased if an assigned cluster is freed.
657                  <1>      ;
658                  <1>      ;
659                  <1>      ; (Modified registers: EAX, EBX, -ECX-, EDX)
660                  <1>
661 0000BBA7 A3[FA400100] <1>      mov     [FAT_CurrentCluster], eax
662 0000BBAC 890D[9C430100] <1>      mov     [ClusterValue], ecx
663                  <1>
664                  <1> loc_update_cluster_check_fat_buffer:
665 0000BBB2 8A1E      <1>      mov     bl, [esi+LD_Name]
666 0000BBB4 381D[FF400100] <1>      cmp     [FAT_BuffDrvName], bl
667 0000BBBA 741A      <1>      je      short loc_update_cluster_check_fat_type
668 0000BBBC 803D[FE400100]02 <1>      cmp     byte [FAT_BuffValidData], 2
669 0000BBC3 0F84C2000000 <1>      jbe     loc_uc_save_fat_buffer
670                  <1>
671                  <1> loc_uc_reset_fat_buffer_validation:
672 0000BBC9 C605[FE400100]00 <1>      mov     byte [FAT_BuffValidData], 0
673                  <1>
674                  <1> loc_uc_check_fat_type_reset_drvname:
675 0000BBD0 881D[FF400100] <1>      mov     [FAT_BuffDrvName], bl
676                  <1>
677                  <1> loc_update_cluster_check_fat_type:
678 0000BBD6 29D2      <1>      sub     edx, edx ; 26/02/2016
679 0000BBD8 8A5E03      <1>      mov     bl, [esi+LD_FATType]
680 0000BBD8 83F802      <1>      cmp     eax, 2
681 0000BBDE 0F82BE000000 <1>      jnb     update_cluster_inv_data
682 0000BBE4 80FB02      <1>      cmp     bl, 2
683 0000BBE7 0F877A010000 <1>      ja      update_fat32_cluster
684                  <1>      ;cmp     bl, 1
685                  <1>      ;jnb     short update_cluster_inv_data
686 0000BBED 8B4E78      <1>      mov     ecx, [esi+LD_Clusters]
687 0000BBF0 41        <1>      inc     ecx
688 0000BBF1 890D[0A410100] <1>      mov     [LastCluster], ecx
689 0000BBF7 39C8      <1>      cmp     eax, ecx ; dword [LastCluster]
690 0000BBF9 0F87A6000000 <1>      ja      return_uc_fat_stc
691                  <1>      ; TRDOS v1 has a FATal bug here !
692                  <1>      ; or bl, bl ; cmp bl, 0

```


336

693		<1>	; jz short update_fat12_cluster
694		<1>	; !! It would destroy FAT12 floppy disk fs here !!
695		<1>	; ('A:' disks of TRDOS v1 operating system project
696		<1>	; had 'singlix fs', so, I could not differ this mistake
697		<1>	; on a drive 'A:')
698	0000BBFF 80FB01	<1>	cmp bl, 1 ; correct comparison is this !
699	0000BC02 0F86A2000000	<1>	jna update_fat12_cluster
700		<1>	
701		<1>	update_fat16_cluster:
702		<1>	pass_uc_fat16_errc:
703		<1>	;sub edx, edx
704	0000BC08 BB00030000	<1>	mov ebx, 300h ;768
705	0000BC0D F7F3	<1>	div ebx
706		<1>	; EAX = Count of 3 FAT sectors
707		<1>	; DX = Cluster offset in FAT buffer
708	0000BC0F 6689D3	<1>	mov bx, dx
709	0000BC12 66D1E3	<1>	shl bx, 1 ; Multiply by 2
710	0000BC15 66BA0300	<1>	mov dx, 3
711	0000BC19 F7E2	<1>	mul edx
712		<1>	; EAX = FAT Sector
713		<1>	; EDX = 0
714		<1>	; EBX = Byte offset in FAT buffer
715	0000BC1B 8A0D[FE400100]	<1>	mov cl, [FAT_BuffValidData]
716	0000BC21 80F902	<1>	cmp cl, 2
717	0000BC24 750A	<1>	jne short loc_uc_check_fat16_buff_sector_load
718		<1>	
719		<1>	loc_uc_check_fat16_buff_sector_save:
720	0000BC26 3B05[02410100]	<1>	cmp eax, [FAT_BuffSector]
721	0000BC2C 755D	<1>	jne short loc_uc_save_fat_buffer
722	0000BC2E EB15	<1>	jmp short loc_update_fat16_cell
723		<1>	
724		<1>	loc_uc_check_fat16_buff_sector_load:
725	0000BC30 80F901	<1>	cmp cl, 1 ; byte [FAT_BuffValidData]
726	0000BC33 0F85FB010000	<1>	jne loc_uc_load_fat_sectors
727	0000BC39 3B05[02410100]	<1>	cmp eax, [FAT_BuffSector]
728	0000BC3F 0F85EF010000	<1>	jne loc_uc_load_fat_sectors
729		<1>	
730		<1>	loc_update_fat16_cell:
731		<1>	loc_update_fat16_buffer:
732	0000BC45 81C3001C0900	<1>	add ebx, FAT_Buffer ; 26/02/2016
733		<1>	;movzx eax, word [ebx]
734	0000BC4B 668B03	<1>	mov ax, [ebx]
735		<1>	; 01/03/2016
736	0000BC4E 89C2	<1>	mov edx, eax ; old value of the cluster
737	0000BC50 A3[FA400100]	<1>	mov [FAT_CurrentCluster], eax
738	0000BC55 8B0D[9C430100]	<1>	mov ecx, [ClusterValue] ; 32 bits
739	0000BC5B 66890B	<1>	mov [ebx], cx ; 16 bits !
740		<1>	
741	0000BC5E C605[FE400100]02	<1>	mov byte [FAT_BuffValidData], 2
742		<1>	
743	0000BC65 6683F802	<1>	cmp ax, 2
744	0000BC69 723A	<1>	jb short return_uc_fat_stc
745	0000BC6B 3B05[0A410100]	<1>	cmp eax, [LastCluster]
746	0000BC71 7732	<1>	ja short return_uc_fat_stc
747		<1>	
748		<1>	loc_fat_buffer_updated:
749		<1>	; 01/03/2016
750	0000BC73 F8	<1>	clc
751		<1>	loc_fat_buffer_stc_1:
752	0000BC74 9C	<1>	pushf
753	0000BC75 21C9	<1>	and ecx, ecx
754	0000BC77 7506	<1>	jnz short loc_fat_buffer_updated_1
755		<1>	
756		<1>	; 01/03/2016
757		<1>	; new value of the cluster = 0 (free)
758		<1>	; increase free(d) cluster count
759	0000BC79 FF05[06410100]	<1>	inc dword [FAT_ClusterCounter]
760		<1>	
761		<1>	loc_fat_buffer_updated_1: ; new value of the cluster > 0
762	0000BC7F 09D2	<1>	or edx, edx ; 02/03/2016
763	0000BC81 7506	<1>	jnz short loc_fat_buffer_updated_2
764		<1>	; old value of the cluster = 0 (it was free cluster)
765		<1>	; decrease free(d) cluster count
766	0000BC83 FF0D[06410100]	<1>	dec dword [FAT_ClusterCounter] ; it may be negative number
767		<1>	
768		<1>	loc_fat_buffer_updated_2:
769	0000BC89 9D	<1>	popf
770	0000BC8A C3	<1>	retn
771		<1>	
772		<1>	loc_uc_save_fat_buffer:
773		<1>	; byte [FAT_BuffValidData] = 2
774	0000BC8B E8D4010000	<1>	call save_fat_buffer
775	0000BC90 0F8297010000	<1>	jc loc_fat_sectors_rw_error2
776		<1>	;mov byte [FAT_BuffValidData], 1
777	0000BC96 A1[FA400100]	<1>	mov eax, [FAT_CurrentCluster]
778		<1>	;mov ecx, [ClusterValue]
779		<1>	;jmp short loc_update_cluster_check_fat_buffer
780	0000BC9B 8A1E	<1>	mov bl, [esi+LD_Name] ; 01/03/2016
781	0000BC9D E927FFFFFF		

```

795                                     <1> pass_uc_fat12_errc:
796                                     <1>         ;sub    edx, edx
797 0000BCAA BB00040000               <1>         mov    ebx, 400h ;1024
798 0000BCAF F7F3                    <1>         div    ebx
799                                     <1>         ; EAX = Count of 3 FAT sectors
800                                     <1>         ; DX = Cluster offset in FAT buffer
801 0000BCB1 66B90300               <1>         mov    cx, 3
802 0000BCB5 6689C3                  <1>         mov    bx, ax
803 0000BCB8 6689C8                  <1>         mov    ax, cx ; 3
804 0000BCBB 66F7E2                  <1>         mul    dx      ; Multiply by 3
805 0000BCBE 66D1E8                  <1>         shr    ax, 1   ; Divide by 2
806 0000BCC1 6693                    <1>         xchg   bx, ax
807                                     <1>         ; EAX = Count of 3 FAT sectors
808                                     <1>         ; EBX = Byte Offset in FAT buffer
809 0000BCC3 66F7E1                  <1>         mul    cx ; 3 * AX
810                                     <1>         ; EAX = FAT Beginning Sector
811                                     <1>         ; EDX = 0
812 0000BCC6 8A0D[FE400100]          <1>         mov    cl, [FAT_BuffValidData]
813                                     <1>         ; TRDOS v1 has a FATal bug here !
814                                     <1>         ; (it does not have 'cmp cl, 2' instruction here !
815                                     <1>         ; while 'jne' is existing !)
816 0000BCCC 80F902                  <1>         cmp    cl, 2 ; 2 = dirty buffer (must be written to disk)
817 0000BCCF 750A                    <1>         jne    short loc_uc_check_fat12_buff_sector_load
818                                     <1>
819                                     <1> loc_uc_check_fat12_buff_sector_save:
820 0000BCD1 3B05[02410100]          <1>         cmp    eax, [FAT_BuffSector]
821 0000BCD7 75B2                    <1>         jne    short loc_uc_save_fat_buffer
822 0000BCD9 EB15                    <1>         jmp     short loc_update_fat12_cell
823                                     <1>
824                                     <1> loc_uc_check_fat12_buff_sector_load:
825 0000BCDB 80F901                  <1>         cmp    cl, 1 ; byte ptr [FAT_BuffValidData]
826 0000BCDE 0F8550010000            <1>         jne    loc_uc_load_fat_sectors
827 0000BCE4 3B05[02410100]          <1>         cmp    eax, [FAT_BuffSector]
828 0000BCEA 0F8544010000            <1>         jne    loc_uc_load_fat_sectors
829                                     <1>
830                                     <1> loc_update_fat12_cell:
831 0000BCF0 81C3001C0900            <1>         add    ebx, FAT_Buffer ; 26/02/2016
832 0000BCF6 668B0D[FA400100]        <1>         mov    cx, [FAT_CurrentCluster]
833 0000BCFD 66D1E9                  <1>         shr    cx, 1
834 0000BD00 668B03                  <1>         mov    ax, [ebx]
835 0000BD03 6689C2                  <1>         mov    dx, ax
836 0000BD06 7344                    <1>         jnc    short uc_fat12_nc_even
837                                     <1>
838 0000BD08 6683E00F                <1>         and    ax, 0Fh
839 0000BD0C 8B0D[9C430100]          <1>         mov    ecx, [ClusterValue] ; 32 bits
840 0000BD12 66C1E104                <1>         shl    cx, 4
841 0000BD16 6609C1                  <1>         or     cx, ax
842 0000BD19 6689D0                  <1>         mov    ax, dx
843 0000BD1C 66890B                  <1>         mov    [ebx], cx ; 16 bits !
844 0000BD1F 66C1E804                <1>         shr    ax, 4 ; al(bit4..7)+ah(bit0..7)
845                                     <1>
846                                     <1> update_fat12_buffer:
847 0000BD23 A3[FA400100]            <1>         mov    [FAT_CurrentCluster], eax
848 0000BD28 89C2                    <1>         mov    edx, eax ; 01/03/2016
849 0000BD2A C605[FE400100]02        <1>         mov    byte [FAT_BuffValidData], 2
850 0000BD31 6683F802                <1>         cmp    ax, 2
851 0000BD35 0F826AFFFFFFFF          <1>         jnb    return_uc_fat_stc
852 0000BD3B 3B05[0A410100]          <1>         cmp    eax, [LastCluster]
853 0000BD41 0F875EFFFFFF            <1>         ja     return_uc_fat_stc
854 0000BD47 E927FFFFFF              <1>         jmp     loc_fat_buffer_updated
855                                     <1>
856                                     <1> uc_fat12_nc_even:
857 0000BD4C 662500F0                <1>         and    ax, 0F000h
858 0000BD50 8B0D[9C430100]          <1>         mov    ecx, [ClusterValue] ; 32 bits
859 0000BD56 80E50F                  <1>         and    ch, 0Fh
860 0000BD59 6609C1                  <1>         or     cx, ax
861 0000BD5C 6689D0                  <1>         mov    ax, dx
862 0000BD5F 66890B                  <1>         mov    [ebx], cx ; 16 bits !
863 0000BD62 80E40F                  <1>         and    ah, 0Fh ; al(bit0..7)+ah(bit0..3)
864 0000BD65 EBBC                    <1>         jmp     short update_fat12_buffer
865                                     <1>
866                                     <1> update_fat32_cluster:
867 0000BD67 8B4E78                  <1>         mov    ecx, [esi+LD_Clusters]
868 0000BD6A 41                      <1>         inc    ecx
869 0000BD6B 890D[0A410100]          <1>         mov    [LastCluster], ecx
870                                     <1>
871 0000BD71 39C8                    <1>         cmp    eax, ecx
872 0000BD73 0F872CFFFFFF            <1>         ja     return_uc_fat_stc
873                                     <1>
874                                     <1> pass_uc_fat32_errc:
875                                     <1>         ;sub    edx, edx
876 0000BD79 BB80010000               <1>         mov    ebx, 180h ;384
877 0000BD7E F7F3                    <1>         div    ebx
878                                     <1>         ; EAX = Count of 3 FAT sectors
879                                     <1>         ; DX = Cluster offset in FAT buffer
880 0000BD80 89D3                    <1>         mov    ebx, edx
881 0000BD82 C1E302                  <1>         shl    ebx, 2 ; Multiply by 4
882 0000BD85 BA03000000              <1>         mov    edx, 3
883 0000BD8A F7E2                    <1>         mul    edx
884                                     <1>         ; EBX = Cluster Offset in FAT buffer
885                                     <1>         ; EAX = FAT Sector
886                                     <1>         ; EDX = 0
887 0000BD8C 8A0D[FE400100]          <1>         mov    cl, [FAT_BuffValidData]
888 0000BD92 80F902                  <1>         cmp    cl, 2
889 0000BD95 750E                    <1>         jne    short loc_uc_check_fat32_buff_sector_load
890                                     <1>
891                                     <1> loc_uc_check_fat32_buff_sector_save:
892 0000BD97 3B05[02410100]          <1>         cmp    eax, [FAT_BuffSector]
893 0000BD9D 0F85E8FEFFFFFF          <1>         jne    loc_uc_save_fat_buffer
894 0000BDA3 EB11                    <1>         jmp     short loc_update_fat32_cell
895                                     <1>
896                                     <1> loc_uc_check_fat32_buff_sector_load:

```

```

897 0000BDA5 80F901      <1>      cmp     cl, 1 ; byte [FAT_BuffValidData]
898 0000BDA8 0F8586000000 <1>      jne     loc_uc_load_fat_sectors
899 0000BDAE 3B05[02410100] <1>      cmp     eax, [FAT_BuffSector]
900 0000BDB4 757E      <1>      jne     loc_uc_load_fat_sectors
901      <1>
902      <1> loc_update_fat32_cell:
903      <1> loc_update_fat32_buffer:
904 0000BDB6 81C3001C0900 <1>      add     ebx, FAT_Buffer ; 26/02/2016
905 0000BDBC 8B03      <1>      mov     eax, [ebx]
906 0000BDBE 25FFFFFFF0F <1>      and     eax, 0FFFFFFFh ; 28 bit cluster value
907      <1>
908 0000BDC3 8B15[FA400100] <1>      mov     edx, [FAT_CurrentCluster] ; 01/03/2016
909      <1>
910 0000BDC9 A3[FA400100] <1>      mov     [FAT_CurrentCluster], eax
911 0000BDCE 8B0D[9C430100] <1>      mov     ecx, [ClusterValue]
912 0000BDD4 890B      <1>      mov     [ebx], ecx ; 29/02/2016
913      <1>
914 0000BDD6 C605[FE400100]02 <1>      mov     byte [FAT_BuffValidData], 2
915      <1>
916      <1> ; 01/03/2016
917 0000BDDD 21C0      <1>      and     eax, eax ; was it free cluster ?
918 0000BDDF 7514      <1>      jnz     short loc_upd_fat32_c0
919      <1>
920      <1> ;or     ecx, ecx ; it will be left free ?!
921      <1> ;jz     short loc_upd_fat32_c3
922      <1>
923 0000BDE1 3B563E      <1>      cmp     edx, [esi+LD_BPB+BPB_Reserved+4] ; First free cluster
924 0000BDE4 7520      <1>      jne     short loc_upd_fat32_c3
925      <1>
926 0000BDE6 3B15[0A410100] <1>      cmp     edx, [LastCluster]
927 0000BDEC 7207      <1>      jb     short loc_upd_fat32_c0
928      <1>
929 0000BDEE BA02000000 <1>      mov     edx, 2 ; rewind !
930 0000BDF3 EB0E      <1>      jmp     short loc_upd_fat32_c2
931      <1>
932      <1> loc_upd_fat32_c0:
933 0000BDF5 FF463E      <1>      inc     dword [esi+LD_BPB+BPB_Reserved+4] ; set it to next cluster
934 0000BDF8 EB0C      <1>      jmp     short loc_upd_fat32_c3
935      <1>
936      <1> loc_upd_fat32_c1:
937 0000BDFA 09C9      <1>      or     ecx, ecx ; will it be free cluster ?
938 0000BDFC 7508      <1>      jnz     short loc_upd_fat32_c3
939      <1>
940 0000BDFE 3B563E      <1>      cmp     edx, [esi+LD_BPB+BPB_Reserved+4] ; First free cluster
941 0000BE01 7303      <1>      jnb     short loc_upd_fat32_c3
942      <1>
943      <1> loc_upd_fat32_c2:
944 0000BE03 89563E      <1>      mov     [esi+LD_BPB+BPB_Reserved+4], edx
945      <1>
946      <1> loc_upd_fat32_c3:
947 0000BE06 89C2      <1>      mov     edx, eax
948      <1>
949      <1> loc_upd_fat32_c4:
950 0000BE08 83F802      <1>      cmp     eax, 2
951 0000BE0B 0F8294FEFFFF <1>      jnb     return_uc_fat_stc
952      <1>
953      <1> pass_uc_fat32_c_zero_check_2:
954 0000BE11 3B05[0A410100] <1>      cmp     eax, [LastCluster]
955 0000BE17 0F8788FEFFFF <1>      ja     return_uc_fat_stc
956      <1>
957 0000BE1D E951FEFFFF <1>      jmp     loc_fat_buffer_updated
958      <1>
959      <1> loc_fat_sectors_rw_error1:
960      <1> ;mov     byte [FAT_BuffValidData], 0
961      <1> ; 23/10/2016 (15h -> 17)
962      <1> ; 23/03/2016
963 0000BE22 B811000000 <1>      mov     eax, 17 ; Drive not ready or read error
964 0000BE27 8825[FE400100] <1>      mov     [FAT_BuffValidData], ah ; 0
965      <1>
966      <1> loc_fat_sectors_rw_error2:
967      <1> ;mov     eax, error code
968      <1> ;mov     edx, 0
969 0000BE2D 8B0D[9C430100] <1>      mov     ecx, [ClusterValue]
970 0000BE33 C3      <1>      retn
971      <1>
972      <1> loc_uc_load_fat_sectors:
973 0000BE34 A3[02410100] <1>      mov     [FAT_BuffSector], eax
974      <1>
975      <1> load_uc_fat_sectors_zero:
976 0000BE39 034660 <1>      add     eax, [esi+LD_FATBegin]
977 0000BE3C BB001C0900 <1>      mov     ebx, FAT_Buffer
978 0000BE41 B903000000 <1>      mov     ecx, 3
979 0000BE46 E88C330000 <1>      call    disk_read
980 0000BE4B 72D5      <1>      jc     short loc_fat_sectors_rw_error1
981      <1>
982 0000BE4D C605[FE400100]01 <1>      mov     byte [FAT_BuffValidData], 1
983 0000BE54 A1[FA400100] <1>      mov     eax, [FAT_CurrentCluster]
984 0000BE59 8B0D[9C430100] <1>      mov     ecx, [ClusterValue]
985 0000BE5F E972FDFFFF <1>      jmp     loc_update_cluster_check_fat_type
986      <1>
987      <1> save_fat_buffer:
988      <1> ; 15/10/2016
989      <1> ; 01/03/2016
990      <1> ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
991      <1> ; 11/08/2011
992      <1> ; 09/02/2005
993      <1> ; INPUT ->
994      <1> ; None
995      <1> ; OUTPUT ->
996      <1> ; cf = 0 -> OK.
997      <1> ; cf = 1 -> error code in AL (EAX)
998      <1> ;

```

```

999      <1>      ;      EBX = FAT_Buffer address
1000     <1>      ;
1001     <1>      ; (EAX, EDX, ECX will be modified)
1002     <1>
1003     <1>      ;cmp    byte [FAT_BuffValidData], 2
1004     <1>      ;je     short loc_save_fat_buff
1005     <1>
1006     <1> ;loc_save_fat_buffer_retn:
1007     <1> ;     xor    eax, eax
1008     <1> ;     retn
1009     <1>
1010     <1> loc_save_fat_buff:
1011     <1>     xor    edx, edx
1012     <1>     mov    dh, [FAT_BuffDrvName]
1013     <1>     cmp    dh, 'A'
1014     <1>     jnb   short loc_save_fat_buffer_inv_data_retn
1015     <1>     sub    dh, 'A'
1016     <1>     push   esi ; *
1017     <1>     mov    esi, Logical_DOSDisks
1018     <1>     add    esi, edx
1019     <1>
1020     <1>     mov    dl, [esi+LD_FATType]
1021     <1>     and    dl, dl
1022     <1>     jz     short loc_save_fat_buffer_inv_data_pop_retn
1023     <1>
1024     <1>     mov    eax, [FAT_BuffSector]
1025     <1>     cmp    dl, 2
1026     <1>     ja     short loc_save_fat32_buff
1027     <1>
1028     <1> loc_save_fat_12_16_buff:
1029     <1>     ; 01/03/2016
1030     <1>     ; TRDOS v1 has a FAtal bug here!
1031     <1>     ; Correct code: mov dx, word ptr [FAT_BuffSector]+2
1032     <1>     ; (DX:AX in TRDOS v1 -> EAX in TRDOS v2)
1033     <1>     ;
1034     <1>     movzx  ecx, word [esi+LD_BPB+FATSecs]
1035     <1>     sub    ecx, eax
1036     <1>     ; TRDOS v1 has a bug here... ('pop esi' was forgotten!)
1037     <1>     ;jna   short loc_save_fat_buffer_inv_data_retn ; wrong addr!
1038     <1>     jna   short loc_save_fat_buffer_inv_data_pop_retn ; correct addr.
1039     <1>     jmp    short loc_save_fat_buffer_check_rs3
1040     <1>
1041     <1> loc_save_fat32_buff:
1042     <1>     mov    ecx, [esi+LD_BPB+FAT32_FAT_Size]
1043     <1>     sub    ecx, eax
1044     <1>     ja     short loc_save_fat_buffer_check_rs3
1045     <1>
1046     <1> loc_save_fat_buffer_inv_data_pop_retn:
1047     <1>     pop    esi ; *
1048     <1> loc_save_fat_buffer_inv_data_retn:
1049     <1>     mov    eax, 0Dh ; Invalid DATA
1050     <1>     retn
1051     <1>
1052     <1> loc_save_fat_buff_remain_sectors_3:
1053     <1>     mov    ecx, 3
1054     <1>     jmp    short loc_save_fat_buff_continue
1055     <1>
1056     <1> loc_save_fat_buffer_check_rs3:
1057     <1>     cmp    ecx, 3
1058     <1>     ja     short loc_save_fat_buff_remain_sectors_3
1059     <1>
1060     <1> loc_save_fat_buff_continue:
1061     <1>     mov    ebx, FAT_Buffer
1062     <1>     add    eax, [esi+LD_FATBegin]
1063     <1>     push   ecx
1064     <1>     call  disk_write
1065     <1>     pop    ecx
1066     <1>     jc     short loc_save_FAT_buff_write_err
1067     <1>
1068     <1>     cmp    byte [esi+LD_FATType], 2
1069     <1>     jna   short loc_calc_2nd_fat12_16_addr
1070     <1>
1071     <1> loc_calc_2nd_fat32_addr:
1072     <1>     mov    eax, [esi+LD_BPB+FAT32_FAT_Size]
1073     <1>     jmp    short loc_calc_2nd_fat_addr
1074     <1>
1075     <1> loc_calc_2nd_fat12_16_addr:
1076     <1>     movzx  eax, word [esi+LD_BPB+FATSecs]
1077     <1>
1078     <1> loc_calc_2nd_fat_addr:
1079     <1>     add    eax, [esi+LD_FATBegin]
1080     <1>     add    eax, [FAT_BuffSector]
1081     <1>     mov    ebx, FAT_Buffer
1082     <1>     ; ecx = 1 to 3
1083     <1>     call  disk_write
1084     <1>     jc     short loc_save_FAT_buff_write_err
1085     <1>     ; Valid buffer (1 = valid but do not save)
1086     <1>     mov    byte [FAT_BuffValidData], 1
1087     <1>
1088     <1> loc_save_FAT_buff_write_err:
1089     <1>     pop    esi ; *
1090     <1>     mov    ebx, FAT_Buffer
1091     <1>     ; 15/10/2016 (1Dh -> 18)
1092     <1>     ; 23/03/2016 (1Dh)
1093     <1>     mov    eax, 18 ; Drive not ready or write error
1094     <1>     retn
1095     <1>
1096     <1> calculate_fat_freespace:
1097     <1>     ; 23/03/2016
1098     <1>     ; 02/03/2016
1099     <1>     ; 01/03/2016
1100     <1>     ; 29/02/2016

```



```

1101      <1>      ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
1102      <1>      ; 30/04/2011
1103      <1>      ; 03/04/2010
1104      <1>      ; 2005
1105      <1>      ; INPUT ->
1106      <1>      ;      EAX = Cluster count to be added or subtracted
1107      <1>      ;      If BH = FFh, ESI = TR-DOS Logical Drive Description Table
1108      <1>      ;      If BH < FFh, BH = TR-DOS Logical Drive Number
1109      <1>      ;      BL:
1110      <1>      ;      0 = Calculate, 1 = Add, 2 = Subtract, 3 = Get (Not Set/Calc)
1111      <1>      ; OUTPUT ->
1112      <1>      ;      EAX = Free Space in sectors
1113      <1>      ;      ESI = Logical Dos Drive Description Table address
1114      <1>      ;      BH = Logical Dos Drive Number (same with input value of BH)
1115      <1>      ;      BL = Type of operation (same with input value of BL)
1116      <1>      ;      ECX = 0 -> valid
1117      <1>      ;      ECX > 0 -> error or invalid
1118      <1>      ;      If EAX = FFFFFFFFh, it is 're-calculation needed'
1119      <1>      ;      sign due to r/w error
1120      <1>
1121      0000BEF9 66891D[A2430100] <1>      mov     [CFS_OPType], bx
1122      0000BF00 A3[A4430100] <1>      mov     [CFS_CC], eax
1123      <1>
1124      0000BF05 80FFFF <1>      cmp     bh, 0FFh
1125      0000BF08 740B <1>      je      short pass_calculate_freespace_get_drive_dt_offset
1126      <1>
1127      <1> loc_calculate_freespace_get_drive_dt_offset:
1128      0000BF0A 31C0 <1>      xor     eax, eax
1129      0000BF0C 88FC <1>      mov     ah, bh
1130      0000BF0E BE00010900 <1>      mov     esi, Logical_DOSDisks
1131      0000BF13 01C6 <1>      add     esi, eax
1132      <1>
1133      <1> pass_calculate_freespace_get_drive_dt_offset:
1134      0000BF15 08DB <1>      or      bl, bl
1135      0000BF17 7435 <1>      jz      short loc_reset_fcc
1136      <1>
1137      <1> loc_get_free_sectors:
1138      0000BF19 8B4674 <1>      mov     eax, [esi+LD_FreeSectors]
1139      <1>
1140      <1>      ;xor     ecx, ecx
1141      <1>      ;dec     ecx ; 0FFFFFFFFh
1142      <1>      ;cmp     eax, ecx ; 29/02/2016
1143      <1>      ;je      short loc_get_free_sectors_retn ; recalculation is needed!
1144      <1>
1145      <1>      ; 23/03/2016
1146      0000BF1C 8B4E70 <1>      mov     ecx, [esi+LD_TotalSectors]
1147      0000BF1F 39C1 <1>      cmp     ecx, eax ; Total sectors must be greater than Free sectors !
1148      0000BF21 7707 <1>      ja      short loc_get_free_sectors_check_optype
1149      <1>
1150      0000BF23 31C0 <1>      xor     eax, eax
1151      0000BF25 48 <1>      dec     eax ; 0FFFFFFFFh ; recalculation is needed!
1152      0000BF26 894674 <1>      mov     [esi+LD_FreeSectors], eax ; reset (for recalculation)
1153      <1>
1154      <1> loc_get_free_sectors_retn:
1155      0000BF29 C3 <1>      retn
1156      <1>
1157      <1> loc_get_free_sectors_check_optype:
1158      0000BF2A 80FB03 <1>      cmp     bl, 3
1159      0000BF2D 7203 <1>      jb      short loc_set_fcc
1160      <1>
1161      0000BF2F 29C9 <1>      sub     ecx, ecx ; 0
1162      <1>
1163      0000BF31 C3 <1>      retn
1164      <1>
1165      <1> loc_set_fcc:
1166      0000BF32 807E0302 <1>      cmp     byte [esi+LD_FATType], 2
1167      0000BF36 0F87DF000000 <1>      ja      loc_update_FAT32_fs_info_fcc
1168      <1>
1169      <1>      ;mov     eax, [esi+LD_FreeSectors]
1170      0000BF3C 0FB64E13 <1>      movzx    ecx, byte [esi+LD_BPB+SecPerClust]
1171      0000BF40 29D2 <1>      sub     edx, edx
1172      0000BF42 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      0000BF44 A3[1B410100] <1>      mov     [FreeClusterCount], eax
1180      0000BF49 E988000000 <1>      jmp     loc_set_free_sectors_FAT12_FAT16
1181      <1>
1182      <1> loc_reset_fcc:
1183      0000BF4E 31C0 <1>      xor     eax, eax
1184      0000BF50 A3[1B410100] <1>      mov     [FreeClusterCount], eax ; 0
1185      0000BF55 8B5678 <1>      mov     edx, [esi+LD_Clusters]
1186      0000BF58 42 <1>      inc     edx
1187      0000BF59 8915[0A410100] <1>      mov     [LastCluster], edx
1188      <1>
1189      0000BF5F 807E0302 <1>      cmp     byte [esi+LD_FATType], 2
1190      0000BF63 7647 <1>      jna      short loc_count_free_fat_clusters_0
1191      <1>
1192      0000BF65 48 <1>      dec     eax ; FFFFFFFFh
1193      0000BF66 A3[AC430100] <1>      mov     [CFS_FAT32FC], eax
1194      <1>
1195      <1>      ; 29/02/2016
1196      0000BF6B 89463A <1>      mov     [esi+LD_BPB+BPB_Reserved], eax ; reset
1197      0000BF6E 89463E <1>      mov     [esi+LD_BPB+BPB_Reserved+4], eax ; reset
1198      <1>
1199      0000BF71 B802000000 <1>      mov     eax, 2
1200      <1>
1201      <1> loc_count_fc_next_cluster_0:
1202      0000BF76 50 <1>      push    eax

```

1203	0000BF77	E801F9FFFF	<1>	call	get_next_cluster
1204	0000BF7C	7310	<1>	jnc	short loc_check_fat32_ff_cluster
1205	0000BF7E	09C0	<1>	or	eax, eax
1206	0000BF80	741E	<1>	jz	short pass_inc_cfs_fcc_0
1207			<1>		
1208			<1>	loc_put_fcc_unknown_sign:	
1209	0000BF82	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	0000BF83	8B15[AC430100]	<1>	mov	edx, [CFS_FAT32FC] ; First Free Cluster
1217			<1>	;	Save First Free Cluster value in FAT32 'BPB_Reserved+4' area
1218	0000BF89	89563E	<1>	mov	[esi+LD_BPB+BPB_Reserved+4], edx
1219			<1>		
1220	0000BF8C	EB7D	<1>	jmp	loc_put_fcc_invalid_sign
1221			<1>		
1222			<1>	loc_check_fat32_ff_cluster:	
1223	0000BF8E	09C0	<1>	or	eax, eax
1224	0000BF90	750E	<1>	jnz	short pass_inc_cfs_fcc_0
1225	0000BF92	58	<1>	pop	eax
1226	0000BF93	A3[AC430100]	<1>	mov	[CFS_FAT32FC], eax
1227			<1>	;mov	dword [FreeClusterCount], 1
1228	0000BF98	FF05[1B410100]	<1>	inc	dword [FreeClusterCount]
1229	0000BF9E	EB27	<1>	jmp	short pass_inc_cfs_fcc_1
1230			<1>		
1231			<1>	pass_inc_cfs_fcc_0:	
1232	0000BFA0	58	<1>	pop	eax
1233			<1>		
1234			<1>	pass_inc_cfs_fcc_0c:	
1235	0000BFA1	40	<1>	inc	eax ; add eax, 1
1236	0000BFA2	3B05[0A410100]	<1>	cmp	eax, [LastCluster]
1237	0000BFA8	76CC	<1>	jna	short loc_count_fc_next_cluster_0
1238	0000BFAA	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	0000BFAC	B002	<1>	mov	al, 2
1243			<1>		
1244			<1>	loc_count_fc_next_cluster:	
1245	0000BFAE	50	<1>	push	eax
1246	0000BFAF	E8C9F8FFFF	<1>	call	get_next_cluster
1247	0000BFB4	720C	<1>	jc	short loc_count_fcc_stc
1248			<1>		
1249			<1>	loc_count_free_clusters_1:	
1250	0000BFB6	21C0	<1>	and	eax, eax
1251	0000BFB8	750C	<1>	jnz	short pass_inc_cfs_fcc
1252			<1>		
1253	0000BFBA	FF05[1B410100]	<1>	inc	dword [FreeClusterCount]
1254	0000BFC0	EB04	<1>	jmp	short pass_inc_cfs_fcc
1255			<1>		
1256			<1>	loc_count_fcc_stc:	
1257	0000BFC2	09C0	<1>	or	eax, eax
1258	0000BFC4	75BC	<1>	jnz	short loc_put_fcc_unknown_sign ; 29/02/2016
1259			<1>		
1260			<1>	pass_inc_cfs_fcc:	
1261	0000BFC6	58	<1>	pop	eax
1262			<1>		
1263			<1>	pass_inc_cfs_fcc_1:	
1264	0000BFC7	40	<1>	inc	eax ; add eax, 1
1265	0000BFC8	3B05[0A410100]	<1>	cmp	eax, [LastCluster]
1266	0000BFCE	76DE	<1>	jna	short loc_count_fc_next_cluster
1267			<1>		
1268			<1>	loc_set_free_sectors:	
1269	0000BFD0	807E0302	<1>	cmp	byte [esi+LD_FATType], 2
1270	0000BFD4	7745	<1>	ja	short loc_update_FAT32_fs_info_fcc
1271			<1>		
1272			<1>	loc_set_free_sectors_FAT12_FAT16:	
1273	0000BFD6	803D[A2430100]00	<1>	cmp	byte [CFS_OPType], 0
1274	0000BFDD	761C	<1>	jna	short pass_FAT_add_sub_fcc
1275	0000BFDF	A1[A4430100]	<1>	mov	eax, [CFS_CC]
1276	0000BFE4	803D[A2430100]01	<1>	cmp	byte [CFS_OPType], 1
1277	0000BFEB	7708	<1>	ja	short pass_FAT_add_f

```

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 0000C01B 803D[A2430100]01 <1> cmp byte [CFS_OPType], 1
1309 0000C022 7221 <1> jnb short loc_cfs_FAT32_get_rcalc_parms ; 0 = recalculated
1310 0000C024 7406 <1> je short loc_check_fcc_FSINFO_op1 ; 1 = add
1311 <1> loc_check_fcc_FSINFO_op2: ; subtract
1312 0000C026 F71D[A4430100] <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 0000C02C 31D2 <1> xor edx, edx ; 0
1316 0000C02E 4A <1> dec edx ; 0FFFFFFFFh
1317 0000C02F 8B463A <1> mov eax, [esi+LD_BPB+BPB_Reserved]
1318 0000C032 39D0 <1> cmp eax, edx
1319 0000C034 73D5 <1> jnb short loc_put_fcc_invalid_sign
1320 0000C036 0305[A4430100] <1> add eax, [CFS_CC] ; free cluster count on disk + current count
1321 0000C03C 72CD <1> jc short loc_put_fcc_invalid_sign
1322 <1>
1323 0000C03E A3[1B410100] <1> mov [FreeClusterCount], eax
1324 0000C043 EB0E <1> jmp short loc_cfs_write_FSINFO_sector
1325 <1>
1326 <1> loc_cfs_FAT32_get_rcalc_parms:
1327 0000C045 8B15[AC430100] <1> mov edx, [CFS_FAT32FC]
1328 0000C04B A1[1B410100] <1> mov eax, [FreeClusterCount]
1329 0000C050 89563E <1> mov [esi+LD_BPB+BPB_Reserved+4], edx ; First Free Cluster
1330 <1> loc_cfs_write_FSINFO_sector:
1331 0000C053 89463A <1> mov [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count
1332 <1> ; 01/03/2016
1333 0000C056 E8AA000000 <1> call set_fat32_fsinfo_sector_parms
1334 0000C05B 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_parms
1342 <1> ;
1343 0000C05D 8B0D[1B410100] <1> mov ecx, [FreeClusterCount]
1344 0000C063 0FB64613 <1> movzx eax, byte [esi+LD_BPB+SecPerClust]
1345 0000C067 F7E1 <1> mul ecx
1346 <1> ; 29/02/2016
1347 0000C069 31C9 <1> xor ecx, ecx ; 0
1348 0000C06B 09D2 <1> or edx, edx ; 0 ?
1349 0000C06D 759C <1> jnz loc_put_fcc_invalid_sign
1350 0000C06F 394670 <1> cmp [esi+LD_TotalSectors], eax ; Volume size in sectors
1351 0000C072 7697 <1> jna short loc_put_fcc_invalid_sign
1352 <1> ;
1353 <1> loc_set_FAT32_free_sectors_ok:
1354 0000C074 31D2 <1> xor edx, edx ; 0
1355 0000C076 EB98 <1> jmp short loc_cfs_retn_parms
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 0000C078 89C1 <1> mov ecx, eax
1377 <1>
1378 0000C07A C705[B4430100]FFFF- <1> mov dword [glc_index], 0FFFFFFFFh ; 22/10/2016
1379 0000C082 FFFF <1>
1380 <1>
1381 0000C084 890D[B0430100] <1> loc_glc_get_next_cluster_1:
1382 <1> mov [glc_prevcluster], ecx
1383 0000C08A FF05[B4430100] <1> ; 22/10/2016
1384 <1> inc dword [glc_index]
1385 <1>
1386 0000C090 E8E8F7FFFF <1> loc_glc_get_next_cluster_2:
1387 <1> call get_next_cluster
1388 <1> ; ecx = current/previous cluster
1389 0000C095 73ED <1> ; eax = next/last cluster
1390 <1> jnc short loc_glc_get_next_cluster_1
1391 0000C097 09C0 <1>
1392 0000C099 7509 <1> or eax, eax
1393 <1> jnz short loc_glc_stc_retn
1394 <1>
1395 0000C09B 89C8 <1> ; ecx = previous cluster
1396 <1> mov eax, ecx
1397 <1>
1398 <1> ; previous cluster becomes last cluster (ecx -> eax)
1399 <1> ; previous of previous cluster becomes previous cluster (ecx)
1400 <1>
1401 0000C09D 8B0D[B0430100] <1> loc_glc_prev_cluster_retn:
1402 0000C0A3 C3 <1> mov ecx, [glc_prevcluster]
1403 <1> retn
1404 <1>
1405 0000C0A4 F5 <1> loc_glc_stc_retn:
<1> cmc ;stc

```

```

1406 0000C0A5 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 0000C0A7 31C9      <1>          xor     ecx, ecx ; mov ecx, 0
1426                  <1>          ;mov     byte [FAT_BuffValidData], 0
1427 0000C0A9 890D[06410100] <1>          mov     [FAT_ClusterCounter], ecx ; 0 ; reset
1428                  <1>
1429                  <1> loc_tcc_unlink_clusters:
1430 0000C0AF 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 0000C0B4 73F9      <1>          jnc short loc_tcc_unlink_clusters
1437                  <1>
1438                  <1> pass_tcc_unlink_clusters:
1439 0000C0B6 A2[BB430100] <1>          mov     byte [TCC_FATErr], al
1440 0000C0BB 803D[FE400100]02 <1>          cmp     byte [FAT_BuffValidData], 2
1441 0000C0C2 750E      <1>          jne     short loc_tcc_calculate_FAT_freespace
1442 0000C0C4 E89BFDFFFF <1>          call    save_fat_buffer
1443 0000C0C9 7307      <1>          jnc     short loc_tcc_calculate_FAT_freespace
1444 0000C0CB A2[BB430100] <1>          mov     byte [TCC_FATErr], al ; Error
1445                  <1>          ;mov     byte [FAT_BuffValidData], 0
1446                  <1>
1447                  <1>          ; 01/03/2016
1448 0000C0D0 EB12      <1>          jmp     short loc_tcc_recalculate_FAT_freespace
1449                  <1>
1450                  <1> loc_tcc_calculate_FAT_freespace:
1451 0000C0D2 A1[06410100] <1>          mov     eax, [FAT_ClusterCounter] ; signed (+-) number
1452 0000C0D7 66BB01FF <1>          mov     bx, 0FF01h ; BH = FFh -> ESI = Dos drv desc. table
1453                  <1>          ; BL = 1 -> add cluster
1454 0000C0DB E819FEFFFF <1>          call    calculate_fat_freespace
1455 0000C0E0 21C9      <1>          and     ecx, ecx ; cx = 0 -> valid free sector count
1456 0000C0E2 7409      <1>          jz     short pass_truncate_cc_recalc_FAT_freespace
1457                  <1>
1458                  <1> loc_tcc_recalculate_FAT_freespace:
1459 0000C0E4 66BB00FF <1>          mov     bx, 0FF00h ; recalculate !
1460 0000C0E8 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 0000C0ED 8B0D[06410100] <1>          mov     ecx, [FAT_ClusterCounter]
1465                  <1>
1466 0000C0F3 803D[BB430100]00 <1>          cmp     byte [TCC_FATErr], 0
1467 0000C0FA 7608      <1>          jna     short loc_tcc_unlink_clusters_retn
1468                  <1>
1469                  <1> loc_tcc_unlink_clusters_error:
1470 0000C0FC 0FB605[BB430100] <1>          movzx   eax, byte [TCC_FATErr]
1471 0000C103 F9        <1>          stc
1472                  <1> loc_tcc_unlink_clusters_retn:
1473 0000C104 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 0000C105 E824000000 <1>          call    get_fat32_fsinfo_sector_parms
1491 0000C10A 7221      <1>          jc     short update_fat32_fsinfo_sector_retn
1492                  <1>
1493 0000C10C 8B463A      <1>          mov     eax, [esi+LD_BPB+BPB_Reserved] ; Free Cluster Count
1494 0000C10F 8B563E      <1>          mov     edx, [esi+LD_BPB+BPB_Reserved+4] ; First free Cluster
1495                  <1>
1496                  <1>          ;mov ebx, DOSBootSectorBuff
1497 0000C112 8983E8010000 <1>          mov     [ebx+488], eax
1498 0000C118 8993EC010000 <1>          mov     [ebx+492], edx
1499                  <1>
1500 0000C11E A1[A8430100] <1>          mov     eax, [CFS_FAT32FSINFOSEC]
1501 0000C123 B901000000 <1>          mov     ecx, 1
1502 0000C128 E89B300000 <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 0000C12D 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 0000C12E 0FB74636 <1> movzx eax, word [esi+LD_BPB+FAT32_FSInfoSec]
1534 0000C132 03466C <1> add eax, [esi+LD_StartSector]
1535 0000C135 A3[A8430100] <1> mov [CFS_FAT32FSINFOSEC], eax
1536 <1>
1537 0000C13A BB[FA3E0100] <1> mov ebx, DOSBootSectorBuff
1538 0000C13F B901000000 <1> mov ecx, 1
1539 0000C144 E88E300000 <1> call disk_read
1540 0000C149 7232 <1> jc short loc_read_FAT32_fsinfo_sec_err
1541 <1>
1542 0000C14B BB[FA3E0100] <1> mov ebx, DOSBootSectorBuff
1543 <1>
1544 0000C150 813B52526141 <1> cmp dword [ebx], 41615252h
1545 0000C156 751E <1> jne short loc_read_FAT32_fsinfo_sec_stc
1546 <1>
1547 0000C158 81BBE4010000727241- <1> cmp dword [ebx+484], 61417272h
1547 0000C161 61 <1>
1548 0000C162 7512 <1> jne short loc_read_FAT32_fsinfo_sec_stc
1549 <1>
1550 0000C164 A1[A8430100] <1> mov eax, [CFS_FAT32FSINFOSEC]
1551 0000C169 8B8BE8010000 <1> mov ecx, [ebx+488] ; free cluster count
1552 0000C16F 8B93EC010000 <1> mov edx, [ebx+492] ; first (next) free cluster
1553 <1>
1554 0000C175 C3 <1> retn
1555 <1>
1556 <1> loc_read_FAT32_fsinfo_sec_stc:
1557 <1> ; 15/10/2016 (0Bh -> 28)
1558 0000C176 B81C000000 <1> mov eax, 28 ; Invalid format!
1559 0000C17B 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 0000C17D B811000000 <1> mov eax, 17 ; Drive not ready or read error
1565 <1>
1566 <1> loc_read_FAT32_fsinfo_sec_stc_retn:
1567 0000C182 29DB <1> sub ebx, ebx ; 0
1568 0000C184 F9 <1> stc
1569 0000C185 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 0000C186 A3[D8440100] <1> mov [FAT_anc_LCluster], eax
1594 <1>
1595 0000C18B E844F9FFFF <1> call get_first_free_cluster
1596 0000C190 720B <1> jc short loc_add_new_cluster_retn
1597 <1> ; EAX >= 2 and EAX < FFFFFFFFh is valid
1598 <1>
1599 0000C192 89C2 <1> mov edx, eax
1600 <1>
1601 0000C194 42 <1> inc edx
1602 <1> ;jnz short loc_add_new_cluster_check_ffc_eax
1603 0000C195 7516 <1> jnz short loc_add_new_cluster_save_fcc
1604 <1>
1605 <1> loc_add_new_cluster_no_disk_space_retn:
1606 0000C197 B827000000 <1> mov eax, 27h ; MSDOS err => insufficient disk space
1607 <1> loc_add_new_cluster_stc_retn:
1608 0000C19C F9 <1> stc

```

```

1609                                     <1> loc_add_new_cluster_retn:
1610 0000C19D 0FB65E13                   <1>     movzx ebx, byte [esi+LD_BPB+SecPerClust]
1611 0000C1A1 8B4E74                     <1>     mov  ecx, [esi+LD_FreeSectors]
1612                                     <1>     ;xor  edx, edx
1613                                     <1>     ;stc
1614 0000C1A4 C3                         <1>     retn
1615                                     <1>
1616                                     <1> loc_anc_invalid_format_stc_retn:
1617 0000C1A5 F9                         <1>     stc
1618                                     <1> loc_add_new_cluster_invalid_format_retn:
1619                                     <1>     ; 15/10/2016 (0Bh -> 28)
1620 0000C1A6 B81C000000                 <1>     mov  eax, 28 ; Invalid format
1621 0000C1AB 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> ;     jnb  short loc_add_new_cluster_invalid_format_retn
1626                                     <1>
1627                                     <1> loc_add_new_cluster_save_fcc:
1628 0000C1AD A3[DC440100]               <1>     mov  [FAT_anc_FFCluster], eax
1629                                     <1>
1630                                     <1>     sub  eax, 2
1631 0000C1B5 0FB65E13                   <1>     movzx ebx, byte [esi+LD_BPB+SecPerClust]
1632 0000C1B9 F7E3                       <1>     mul  ebx
1633 0000C1BB 09D2                       <1>     or   edx, edx
1634 0000C1BD 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 0000C1BF 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 0000C1C0 BF00000700                 <1>     mov  edi, Cluster_Buffer ; 70000h (for current TRDOS 386 version)
1651 0000C1C5 89D9                       <1>     mov  ecx, ebx ; sector count
1652 0000C1C7 C1E107                     <1>     shl  ecx, 7 ; 1 sector = 512 bytes -> 128 double words
1653                                     <1>     ;xor  eax, eax ; 0
1654 0000C1CA 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 0000C1CC 89D0                       <1>     mov  eax, edx
1660 0000C1CE 034668                     <1>     add  eax, [esi+LD_DATABegin]
1661 0000C1D1 72D3                       <1>     jc   short loc_add_new_cluster_invalid_format_retn
1662                                     <1>
1663                                     <1>     mov  ecx, ebx ; ECX = sectors per cluster (<256)
1664 0000C1D5 BB00000700                 <1>     mov  ebx, Cluster_Buffer
1665 0000C1DA E8E92F0000                 <1>     call disk_write
1666 0000C1DF 7307                       <1>     jnc  short loc_add_new_cluster_update_fat_nlc
1667                                     <1>
1668                                     <1>     ; 15/10/2016 (1Dh -> 18)
1669 0000C1E1 B812000000                 <1>     mov  eax, 18 ; Write Error
1670 0000C1E6 EBB4                       <1>     jmp  short loc_add_new_cluster_stc_retn
1671                                     <1>
1672                                     <1> loc_add_new_cluster_update_fat_nlc:
1673 0000C1E8 A1[DC440100]               <1>     mov  eax, [FAT_anc_FFCluster]
1674 0000C1ED 31C9                       <1>     xor  ecx, ecx
1675 0000C1EF 890D[06410100]             <1>     mov  [FAT_ClusterCounter], ecx ; 0 ; reset
1676 0000C1F5 49                         <1>     dec  ecx ; 0FFFFFFFh
1677 0000C1F6 E8ACF9FFFF                 <1>     call update_cluster
1678 0000C1FB 7304                       <1>     jnc  short loc_add_new_cluster_update_fat_plc
1679 0000C1FD 09C0                       <1>     or   eax, eax ;EAX = 0 -> cluster value is 0 or eocc
1680 0000C1FF 759B                       <1>     jnz  short loc_add_new_cluster_stc_retn
1681                                     <1>
1682                                     <1> loc_add_new_cluster_update_fat_plc:
1683 0000C201 A1[D8440100]               <1>     mov  eax, [FAT_anc_LCluster]
1684 0000C206 8B0D[DC440100]             <1>     mov  ecx, [FAT_anc_FFCluster]
1685 0000C20C E896F9FFFF                 <1>     call update_cluster
1686 0000C211 7314                       <1>     jnc  short loc_add_new_cluster_save_fat_buffer
1687 0000C213 09C0                       <1>     or   eax, eax ; EAX = 0 -> cluster value is 0 or eocc
1688 0000C215 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>     ;jnb  short loc_add_new_cluster_retn
1693                                     <1>
1694 0000C217 66BB00FF                   <1>     mov  bx, 0FF00h ; recalculate free space (BL = 0)
1695                                     <1>     ; (BH = FFh -> Use ESI as Drv Param. Tbl.)
1696 0000C21B 50                         <1>     push eax
1697 0000C21C E8D8FCFFFF                 <1>     call calculate_fat_freespace
1698 0000C221 58                         <1>     pop  eax
1699 0000C222 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 0000C227 E838FCFFFF                 <1>     call save_fat_buffer
1706 0000C22C 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 0000C22E A1[06410100]               <1>     mov  eax, [FAT_ClusterCounter]

```

```

1711 0000C233 66BB01FF      <1>      mov     bx, 0FF01h ; BH = FFh -> ESI -> Dos drv desc. table
1712                        <1>      ; BL = 1 -> add cluster
1713 0000C237 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 0000C239 E8BBFCFFFF    <1>      call    calculate_fat_freespace
1717                        <1>      ;ECX = 0 -> no error, ECX > 0 -> error or invalid return
1718 0000C23E 21C9          <1>      and     ecx, ecx ; ECX = 0 -> valid free sector count
1719 0000C240 7409          <1>      jz      short loc_add_new_cluster_return_cluster_number
1720                        <1>
1721                        <1> loc_add_new_cluster_recalc_FAT_freespace:
1722 0000C242 66BB00FF    <1>      mov     bx, 0FF00h ; recalculate free space
1723 0000C246 E8AEFCFFFF    <1>      call    calculate_fat_freespace
1724                        <1>      ; cf = 0
1725                        <1> loc_add_new_cluster_return_cluster_number:
1726 0000C24B 89C1          <1>      mov     ecx, eax ; Free sector count
1727 0000C24D A1[DC440100]  <1>      mov     eax, [FAT_anc_FFCluster]
1728 0000C252 0FB65E13    <1>      movzx   ebx, byte [esi+LD_BPB+SecPerClust]
1729                        <1>      ;mov     edi, Cluster_Buffer
1730 0000C256 31D2          <1>      xor     edx, edx
1731 0000C258 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 0000C259 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 0000C25D 807E0300    <1>      cmp     byte [esi+LD_FATType], 0
1755 0000C261 761C          <1>      jna     short write_fs_cluster
1756                        <1>
1757                        <1> write_fat_file_sectors:
1758 0000C263 83E802      <1>      sub     eax, 2 ; Beginning cluster number is always 2
1759 0000C266 0FB65613    <1>      movzx   edx, byte [esi+LD_BPB+BPB_SecPerClust] ; 18/03/2016
1760 0000C26A F7E2          <1>      mul     edx
1761 0000C26C 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 0000C26F E8542F0000    <1>      call    disk_write
1770 0000C274 7306          <1>      jnc     short wclust_retn
1771                        <1>
1772                        <1>      ; 15/10/2016 (1Dh -> 18)
1773 0000C276 B812000000    <1>      mov     eax, 18 ; Drive not ready or write error !
1774 0000C27B C3            <1>      retn
1775                        <1>
1776                        <1> wclust_retn:
1777 0000C27C 29C0          <1>      sub     eax, eax ; 0
1778 0000C27E 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 0000C27F B980000000    <1>      mov     ecx, 128 ; maximum count of sectors (before eof)
1793 0000C284 E801000000    <1>      call    write_fs_sectors
1794 0000C289 C3            <1>      retn
1795                        <1>
1796                        <1> write_fs_sectors:
1797                        <1>      ; 21/03/2016 (TRDOS 386 = TRDOS v2.0)
1798 0000C28A F9            <1>      stc
1799 0000C28B 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      <1>      cmp     byte [esi+LD_FATType], 1
1816      <1>      jnb     short get_fs_section_by_index
1817      <1>
1818      <1>      cmp     ecx, [esi+LD_Clusters]
1819      <1>      jnb     short gcbi_1
1820      <1> gcbi_0:
1821      <1>      stc
1822      <1>      mov     eax, 23h ; Cluster not available !
1823      <1>      ; MSDOS error code: FCB unavailable
1824      <1>      retn
1825      <1> gcbi_1:
1826      <1>      push    ecx
1827      <1>      call    get_next_cluster
1828      <1>      pop     ecx
1829      <1>      jc     short gcbi_3
1830      <1>      loop   gcbi_1
1831      <1> gcbi_2:
1832      <1>      retn
1833      <1> gcbi_3:
1834      <1>      or      eax, eax
1835      <1>      jz      short gcbi_0
1836      <1>      cmc     ; stc
1837      <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      <1>      mov     eax, 0FFFFFFFFh
1854      <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      <1>      mov     eax, 0
1872      <1>      mov     edx, 0
1873      <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: 24/03/2017
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> ; ul.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 0000C2C1 368925[A44E0100] <1>      mov      [ss:u.sp], esp ; Kernel stack points to return address
54      <1>
55      <1>      ; save user registers
56 0000C2C8 1E <1>      push    ds
57 0000C2C9 06 <1>      push    es
58 0000C2CA 0FA0 <1>      push    fs
59 0000C2CC 0FA8 <1>      push    gs
60 0000C2CE 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 0000C2CF 50 <1>      push    eax ; 01/07/2015
67 0000C2D0 66B81000 <1>      mov     ax, KDATA
68 0000C2D4 8ED8 <1>      mov     ds, ax
69 0000C2D6 8EC0 <1>      mov     es, ax
70 0000C2D8 8EE0 <1>      mov     fs, ax
71 0000C2DA 8EE8 <1>      mov     gs, ax
72 0000C2DC A1[20380100] <1>      mov     eax, [k_page_dir]
73 0000C2E1 0F22D8 <1>      mov     cr3, eax
74 0000C2E4 58 <1>      pop     eax ; 01/07/2015
75      <1>      ; 19/10/2015
76 0000C2E5 FC <1>      cld
77      <1>      ;
78 0000C2E6 FE05[A14E0100] <1>      inc     byte [sysflg]
79      <1>      ; incb sysflg / indicate a system routine is in progress
80 0000C2EC FB <1>      sti     ; 18/01/2014
81 0000C2ED 0F851EA0FFFF <1>      jnz     panic ; 24/05/2013
82      <1>      ; beq lf
83      <1>      ; jmp panic ; / called if trap inside system
84      <1> ;1:
85      <1>      ; 17/03/2017
86 0000C2F3 80642438FE <1>      and     byte [esp+ESPACE+8], ~1 ; clear carry flag
87      <1>
88      <1>      ; 16/04/2015
89 0000C2F8 A3[AC4E0100] <1>      mov     [u.r0], eax
90 0000C2FD 8925[A84E0100] <1>      mov     [u.usp], esp ; kernel stack points to user's registers
91      <1>
92      <1>      ; 13/01/2017 (TRDOS 386 Feaure only !)
93 0000C303 803D[1C4F0100]00 <1>      cmp     byte [u.t_lock], 0 ; timer interrupt lock ?
94 0000C30A 0F8799010000 <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 0000C310 C1E002 <1>      shl     eax, 2
116     <1>      ; asl r0 / multiply by 2 to jump indirect in bytes
117 0000C313 3DB4000000 <1>      cmp     eax, end_of_syscalls - syscalls
118     <1>      ; cmp r0,$2f-1f / limit of table (35) exceeded
119     <1>      ; jnb
120     <1>      ; bhis badsys / yes, bad system call
121 0000C318 F5 <1>      cmc
122 0000C319 9C <1>      pushf
123 0000C31A 50 <1>      push    eax
124 0000C31B 8B2D[A44E0100] <1>      mov     ebp, [u.sp] ; Kernel stack at the beginning of sys call
125 0000C321 B0FE <1>      mov     al, 0FEh ; 11111110b
126 0000C323 1400 <1>      adc     al, 0 ; al = al + cf
127 0000C325 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 0000C328 5D <1>      pop     ebp ; eax
131 0000C329 9D <1>      popf
132 0000C32A 0F8204020000 <1>      jc      badsys
133 0000C330 A1[AC4E0100] <1>      mov     eax, [u.r0]
134     <1>      ; system call registers: EAX, EDX, ECX, EBX, ESI, EDI
135 0000C335 FFA5[3BC30000] <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 0000C33B [05E70000] <1>      dd sysver ; 0 ; Get TRDOS 386 version number (v2.0)
152 0000C33F [96C50000] <1>      dd sysexit ; / 1
153 0000C343 [4CC70000] <1>      dd sysfork ; / 2
154 0000C347 [80CB0000] <1>      dd sysread ; / 3
155 0000C34B [9FCB0000] <1>      dd syswrite ; / 4
156 0000C34F [36C90000] <1>      dd sysopen ; / 5
157 0000C353 [57CB0000] <1>      dd sysclose ; / 6
158 0000C357 [CEC60000] <1>      dd syswait ; / 7
159 0000C35B [64C80000] <1>      dd syscreat ; / 8
160 0000C35F [5CD70000] <1>      dd syslink ; / 9
161 0000C363 [A7D70000] <1>      dd sysunlink ; / 10
162 0000C367 [34D80000] <1>      dd sysexec ; / 11
163 0000C36B [2BDC0000] <1>      dd syschdir ; / 12
164 0000C36F [0ADD0000] <1>      dd systime ; / 13
165 0000C373 [19CB0000] <1>      dd sysmkdir ; / 14
166 0000C377 [7DDC0000] <1>      dd syschmod ; / 15
167 0000C37B [DADC0000] <1>      dd syschown ; / 16
168 0000C37F [3DDD0000] <1>      dd sysbreak ; / 17
169 0000C383 [DBD90000] <1>      dd sysstat ; / 18
170 0000C387 [82DD0000] <1>      dd sysseek ; / 19
171 0000C38B [94DD0000] <1>      dd systell ; / 20
172 0000C38F [7DDE0000] <1>      dd sysmount ; / 21
173 0000C393 [91DE0000] <1>      dd sysumount ; / 22
174 0000C397 [06DE0000] <1>      dd syssetuid ; / 23
175 0000C39B [37DE0000] <1>      dd sysgetuid ; / 24
176 0000C39F [19DD0000] <1>      dd sysstime ; / 25
177 0000C3A3 [FADD0000] <1>      dd sysquit ; / 26
178 0000C3A7 [EEDD0000] <1>      dd sysintr ; / 27
179 0000C3AB [EFD90000] <1>      dd sysfstat ; / 28
180 0000C3AF [36CC0000] <1>      dd sysemu ; / 29
181 0000C3B3 [E7CD0000] <1>      dd sysmdate ; / 30
182      <1>      ;dd sysstty ; / 31
183 0000C3B7 [FBCE0000] <1>      dd sysvideo ; 31 ; TRDOS 386 Video Functions (16/05/2016)
184      <1>      ;dd sysgtty ; / 32
185 0000C3BB [4AD70000] <1>      dd sysaudio ; 32 ; TRDOS 386 Audio Functions (16/05/2016)
186      <1>      ;dd sysilgins; / 33
187 0000C3BF [4FCC0000] <1>      dd systimer ; 33 ; TRDOS 386 Timer Functions (18/05/2016)
188 0000C3C3 [A5DE0000] <1>      dd sysssleep ; 34 ; Retro UNIX 8086 v1 feature only !
189      <1>      ; 11/06/2014
190 0000C3C7 [D4DE0000] <1>      dd sysmsg ; 35 ; Retro UNIX 386 v1 feature only !
191      <1>      ; 01/07/2015
192 0000C3CB [AADF0000] <1>      dd sysgeterr ; 36 ; Retro UNIX 386 v1 feature only !
193      <1>      ; 21/09/2015 - get last error number
194 0000C3CF [95F10000] <1>      dd sysfpstat ; 37 ; TRDOS 386 FPU state option (28/02/2017)
195 0000C3D3 [23E70000] <1>      dd syspri ; 38 ; change priority - TRDOS 386 (20/05/2016)
196 0000C3D7 [A9C40000] <1>      dd sysrele ; 39 ; TRDOS 386 (19/05/2016) (0 -> 39)
197 0000C3DB [56E80000] <1>      dd sysfff ; 40 ; Find First File - TRDOS 386 (15/10/2016)
198 0000C3DF [35E90000] <1>      dd sysfnf ; 41 ; Find Next File - TRDOS 386 (15/10/2016)
199 0000C3E3 [A5EF0000] <1>      dd sysalloc ; 42 ; Allocate contiguous memory block/pages
200      <1>      ; TRDOS 386 (19/02/2017) DMA buff fuctions
201 0000C3E7 [54F00000] <1>      dd sysdalloc ; 43 ; Deallocate contiguous memory block/pages
202      <1>      ; TRDOS 386 (19/02/2017) DMA buff fuctions
203 0000C3EB [8FF00000] <1>      dd syscalbac ; 44 ; IRQ Callback and Signal Response Byte
204      <1>      ; service setup - TRDOS 386 (20/02/2017)
205      <1>
206      <1> end_of_syscalls:
207      <1>
208      <1> error:
209      <1>      ; 18/05/2016
210      <1>      ; 13/05/2016
211      <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
212      <1>      ; 16/04/2015 - 17/09/2015 (Retro UNIX 386 v1)
213      <1>      ; 10/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
214      <1>      ;
215      <1>      ; 'error' merely sets the error bit off the processor status (c-bit)
216      <1>      ; then falls right into the 'sysret', 'sysrele' return sequence.
217      <1>      ;
218      <1>      ; INPUTS -> none
219      <1>      ; OUTPUTS ->
220      <1>      ; processor status - carry (c) bit is set (means error)
221      <1>      ;
222      <1>      ; 26/05/2013 (Stack pointer must be reset here!
223      <1>      ; Because, jumps to error procedure
224      <1>      ; disrupts push-pop nesting balance)
225      <1>      ;
226 0000C3EF 8B2D[A44E0100] <1>      mov ebp, [u.sp] ; interrupt (system call) return (iretd) address
227 0000C3F5 804D0801 <1>      or byte [ebp+8], 1 ; set carry bit of flags register
228      <1>      ; (system call will return with cf = 1)
229      <1>      ; bis $1,20.(r1) / set c bit in processor status word below
230      <1>      ; / users stack
231      <1>      ; 17/09/2015
232 0000C3F9 83ED30 <1>      sub ebp, ESPACE ; 48 ; total size of stack frame ('sysdefs.inc')
233      <1>      ; for saving/restoring user registers
234      <1>      ;cmp ebp, [u.usp]
235      <1>      ;je short err0
236 0000C3FC 892D[A84E0100] <1>      mov [u.usp], ebp
237      <1> ;err0:
238      <1>      ; 01/09/2015
239 0000C402 8B25[A84E0100] <1>      mov esp, [u.usp] ; Retro Unix 8086 v1 modification!
240      <1>      ; 10/04/2013
241      <1>      ; (If an I/O error occurs during disk I/O,
242      <1>      ; related procedures will jump to 'error'
243      <1>      ; procedure directly without returning to
244      <1>      ; the caller procedure. So, stack pointer

```

```

245                                     <1>                                     ; must be restored here.)
246                                     <1>                                     ; 13/05/2016
247                                     <1>                                     ; NOTE: (The last) error code is in 'u.error', it can be retrieved by
248                                     <1>                                     ; 'get last error' system call later.
249                                     <1>
250                                     <1>                                     ; 03/09/2015 - 09/06/2015 - 07/08/2013
251 0000C408 C605[0E4F0100]00 <1>                                     mov     byte [u.kcall], 0 ; namei_r, mkdir_w reset
252                                     <1>
253                                     <1> sysret: ; < return from system call>
254                                     <1>                                     ; 01/03/2017
255                                     <1>                                     ; 28/02/2017
256                                     <1>                                     ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
257                                     <1>                                     ; 16/04/2015 - 10/09/2015 (Retro UNIX 386 v1)
258                                     <1>                                     ; 10/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
259                                     <1>                                     ;
260                                     <1>                                     ; 'sysret' first checks to see if process is about to be
261                                     <1>                                     ; terminated (u.bsys). If it is, 'sysexit' is called.
262                                     <1>                                     ; If not, following happens:
263                                     <1>                                     ; 1) The user's stack pointer is restored.
264                                     <1>                                     ; 2) r1=0 and 'iget' is called to see if last mentioned
265                                     <1>                                     ; i-node has been modified. If it has, it is written out
266                                     <1>                                     ; via 'ppoke'.
267                                     <1>                                     ; 3) If the super block has been modified, it is written out
268                                     <1>                                     ; via 'ppoke'.
269                                     <1>                                     ; 4) If the dismountable file system's super block has been
270                                     <1>                                     ; modified, it is written out to the specified device
271                                     <1>                                     ; via 'ppoke'.
272                                     <1>                                     ; 5) A check is made if user's time quantum (uquant) ran out
273                                     <1>                                     ; during his execution. If so, 'tswap' is called to give
274                                     <1>                                     ; another user a chance to run.
275                                     <1>                                     ; 6) 'sysret' now goes into 'sysrele'.
276                                     <1>                                     ; (See 'sysrele' for conclusion.)
277                                     <1>                                     ;
278                                     <1>                                     ; Calling sequence:
279                                     <1>                                     ; jump table or 'br sysret'
280                                     <1>                                     ; Arguments:
281                                     <1>                                     ; -
282                                     <1>                                     ; .....
283                                     <1>                                     ;
284                                     <1>                                     ; ((AX=r1 for 'iget' input))
285                                     <1>                                     ;
286 0000C40F 31C0 <1>                                     xor     eax, eax ; 28/02/2017
287                                     <1> sysret0: ; 29/07/2015 (eax = 0, jump from sysexec)
288 0000C411 FEC0 <1>                                     inc     al ; 04/05/2013
289 0000C413 3805[FA4E0100] <1>                                     cmp     [u.bsys], al ; 1
290                                     <1>                                     ; tstb u.bsys / is a process about to be terminated because
291 0000C419 0F8377010000 <1>                                     jnb     sysexit ; 04/05/2013
292                                     <1>                                     ; bne sysexit / of an error? yes, go to sysexit
293                                     <1> ;mov     esp, [u.usp] ; 24/05/2013 (that is not needed here)
294                                     <1>                                     ; mov u.sp,sp / no point stack to users stack
295 0000C41F FEC8 <1>                                     dec     al ; mov ax, 0
296                                     <1>                                     ; clr r1 / zero r1 to check last mentioned i-node
297 0000C421 E8982D0000 <1>                                     call    iget
298                                     <1>                                     ; jsr r0,iget / if last mentioned i-node has been modified
299                                     <1>                                     ; / it is written out
300                                     <1>                                     ; 10/01/2017
301                                     <1>                                     ; 09/01/2017
302                                     <1> ;sysrele: ; < release >
303                                     <1>                                     ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
304                                     <1>                                     ; 16/04/2015 - 14/10/2015 (Retro UNIX 386 v1)
305                                     <1>                                     ; 10/04/2013 - 07/03/2014 (Retro UNIX 8086 v1)
306                                     <1>                                     ;
307                                     <1>                                     ; 'sysrele' first calls 'tswap' if the time quantum for a user is
308                                     <1>                                     ; zero (see 'sysret'). It then restores the user's registers and
309                                     <1>                                     ; turns off the system flag. It then checked to see if there is
310                                     <1>                                     ; an interrupt from the user by calling 'isintr'. If there is,
311                                     <1>                                     ; the output gets flashed (see isintr) and interrupt action is
312                                     <1>                                     ; taken by a branch to 'intract'. If there is no interrupt from
313                                     <1>                                     ; the user, a rti is made.
314                                     <1>                                     ;
315                                     <1>                                     ; Calling sequence:
316                                     <1>                                     ; Fall through a 'bne' in 'sysret' & ?
317                                     <1>                                     ; Arguments:
318                                     <1>                                     ; -
319                                     <1>                                     ; .....
320                                     <1>                                     ;
321                                     <1>                                     ; 23/02/2014 (swapret)
322                                     <1>                                     ; 22/09/2013
323                                     <1> sysrel0: ;1:
324 0000C426 803D[F04E0100]00 <1>                                     cmp     byte [u.quant], 0 ; 16/05/2013
325                                     <1>                                     ; tstb uquant / is the time quantum 0?
326 0000C42D 7705 <1>                                     ja      short swapret
327                                     <1>                                     ; bne lf / no, don't swap it out
328                                     <1> sysrelease: ; 07/12/2013 (jump from 'clock')
329 0000C42F E899210000 <1>                                     call    tswap
330                                     <1>                                     ; jsr r0,tswap / yes, swap it out
331                                     <1>
332                                     <1> ; Retro Unix 8086 v1 feature: return from 'swap' to 'swapret' address.
333                                     <1> swapret: ;1:
334                                     <1>                                     ; 10/09/2015
335                                     <1>                                     ; 01/09/2015
336                                     <1>                                     ; 14/05/2015
337                                     <1>                                     ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit, pm modifications)
338                                     <1>                                     ; 26/05/2013 (Retro UNIX 8086 v1)
339                                     <1>                                     ; cli
340                                     <1>                                     ; 24/07/2015
341                                     <1>                                     ;
342                                     <1>                                     ; ; 'esp' must be already equal to '[u.usp]' here !
343                                     <1>                                     ; ; mov esp, [u.usp]
344                                     <1>
345                                     <1>                                     ; 22/09/2013
346 0000C434 E8862D0000 <1>                                     call    isintr

```

```

347          <1>      ; 20/10/2013
348 0000C439 7405      <1>      jz      short sysrel1
349 0000C43B E83F010000 <1>      call   intract
350          <1>      ; jsr r0,isintr / is there an interrupt from the user
351          <1>      ;      br intract / yes, output gets flushed, take interrupt
352          <1>      ; / action
353          <1> sysrel1:
354 0000C440 FA          <1>      cli      ; 14/10/2015
355          <1> sysrel2:
356          <1>      ; 28/02/2017
357          <1>      ; Check if there is a (delayed) callback for current user/process
358 0000C441 A0[1F4F0100] <1>      mov     al, [u.irqwait]
359 0000C446 240F        <1>      and     al, 0Fh ; is there a waiting IRQ callback service ?
360 0000C448 7444        <1>      jz      short sysrel8 ; no
361          <1>
362          <1>      ; Set return to IRQ callback service and return from the service
363 0000C44A 0FB6D8      <1>      movzx   ebx, al
364 0000C44D 883D[1F4F0100] <1>      mov     [u.irqwait], bh ; 0 ; reset
365 0000C453 8A9B[DC030100] <1>      mov     bl, [ebx+IRQenum] ; (available) IRQ index +1 (1 to 9)
366          <1>      ; 01/03/2017
367 0000C459 FECB        <1>      dec     bl ; IRQ index number, 0 to 8
368 0000C45B 7831        <1>      js      short sysrel8 ; 0 -> FFh (not in use!?)
369          <1>      ;
370 0000C45D A0[FB4E0100] <1>      mov     al, [u.uno] ; current process (user) number
371 0000C462 3883[F84A0100] <1>      cmp     [ebx+IRQ.owner], al
372 0000C468 7524        <1>      jne     short sysrel8 ; it is not the current user/process !?
373 0000C46A F683[0A4B0100]01 <1>      test    byte [ebx+IRQ.method], 1 ; callback ?
374 0000C471 741B        <1>      jz      short sysrel8 ; not a callback method !?
375          <1>
376 0000C473 8B93[1C4B0100] <1>      mov     edx, [ebx+IRQ.addr] ; IRQ callback service address (virtual)
377 0000C479 C605[204F0100]01 <1>      mov     byte [u.r_lock], 1 ; IRQ callback service in progress flag
378          <1>
379 0000C480 E8F0210000    <1>      call    wswap ; save user's registers & status
380          <1>      ;      (for return from IRQ callback service)
381          <1>
382 0000C485 8B2D[A44E0100] <1>      mov     ebp, [u.sp]; kernel's stack, points to EIP (user)
383 0000C48B 895500        <1>      mov     [ebp], edx ; IRQ call back service address
384          <1> sysrel8:
385 0000C48E FE0D[A14E0100] <1>      dec     byte [sysflg]
386          <1>      ; decb sysflg / turn system flag off
387          <1>
388 0000C494 A1[004F0100] <1>      mov     eax, [u.pgdir]
389 0000C499 0F22D8        <1>      mov     cr3, eax ; 1st PDE points to Kernel Page Table 0 (1st 4 MB)
390          <1>      ; (others are different than kernel page tables)
391          <1>      ; 10/09/2015
392 0000C49C 61          <1>      popad ; edi, esi, ebp, temp (increment esp by 4), ebx, edx, ecx, eax
393          <1>      ; mov (sp)+,sc / restore user registers
394          <1>      ; mov (sp)+,mq
395          <1>      ; mov (sp)+,ac
396          <1>      ; mov (sp)+,r5
397          <1>      ; mov (sp)+,r4
398          <1>      ; mov (sp)+,r3
399          <1>      ; mov (sp)+,r2
400          <1>      ;
401 0000C49D A1[AC4E0100] <1>      mov     eax, [u.r0] ; ((return value in EAX))
402 0000C4A2 0FA9          <1>      pop     gs
403 0000C4A4 0FA1          <1>      pop     fs
404 0000C4A6 07           <1>      pop     es
405 0000C4A7 1F           <1>      pop     ds
406          <1>      ;or word [esp+8], 200h ; 22/01/2017 ; force enabling interrupts
407 0000C4A8 CF          <1>      iretd
408          <1>      ; rti / no, return from interrupt
409          <1>
410          <1> sysrele:
411          <1>      ; 24/03/2017
412          <1>      ; 28/02/2017
413          <1>      ; 27/02/2017
414          <1>      ; 29/01/2017
415          <1>      ; 14/01/2017
416          <1>      ; 13/01/2017
417          <1>      ; 09/01/2017, 10/01/2017, 12/01/2017
418          <1>      ; Major modification for TRDOS 386 (CallBack return)
419          <1>      ;
420          <1>      ; 'sysrele' system call restores previously saved
421          <1>      ; registers and addresses of the process
422          <1>      ; (Main purpose -in TRDOS 386- is to return from
423          <1>      ; timer callback service routine in ring 3 -user mode-.)
424          <1>      ;
425          <1>      ; check if the process is in timer callback phase
426 0000C4A9 803D[1C4F0100]00 <1>      cmp     byte [u.t_lock], 0 ; TIMER INT LOCK
427          <1>      ;je short sysrel0 ; classic (Retro UNIX 386 type) sysrele
428 0000C4B0 7734          <1>      ja      short sysrel3
429          <1>      ; 27/02/2017
430 0000C4B2 803D[204F0100]00 <1>      cmp     byte [u.r_lock], 0 ; IRQ callback lock
431 0000C4B9 0F8667FFFFFF <1>      jna     sysrel0 ; classic sysrele ; 24/03/2017
432 0000C4BF E859000000    <1>      call    sysrel7
433 0000C4C4 803D[204F0100]00 <1>      cmp     byte [u.r_lock], 0 ; IRQ callback service lock
434 0000C4CB 7628          <1>      jna     short sysrel4
435 0000C4CD C605[204F0100]00 <1>      mov     byte [u.r_lock], 0 ; reset
436          <1>      ;mov byte [u.irqwait], 0 ; reset ; 28/02/2017
437 0000C4D4 A0[214F0100] <1>      mov     al, [u.r_mode]
438 0000C4D9 08C0          <1>      or      al, al
439 0000C4DB 7518          <1>      jnz     short sysrel4
440 0000C4DD FEC8          <1>      dec     al
441 0000C4DF A2[214F0100] <1>      mov     [u.r_mode], al ; 0FFh ; not necessary !?
442 0000C4E4 EB32          <1>      jmp     short sysrel6
443          <1> sysrel3:
444          <1>      ; 27/02/2017
445 0000C4E6 E832000000    <1>      call    sysrel7
446          <1>      ; 14/01/2017
447 0000C4EB 28C0          <1>      sub     al, al
448 0000C4ED 3805[1C4F0100] <1>      cmp     [u.t_lock], al ; 0 ; TIMER INT LOCK

```



```

449 0000C4F3 770E      <1>      ja      short sysrel5 ; yes
450                  <1> sysrel4:
451                  <1>      ; 29/01/2017
452 0000C4F5 8B44241C   <1>      mov     eax, [esp+28] ; eax
453 0000C4F9 A3[AC4E0100] <1>      mov     [u.r0], eax
454 0000C4FE E93EFFFFFF   <1>      jmp     sysrel2
455                  <1> sysrel5:
456 0000C503 A2[1C4F0100]   <1>      mov     [u.t_lock], al ; 0 ; reset
457 0000C508 A0[1D4F0100]   <1>      mov     al, [u.t_mode]
458 0000C50D 20C0      <1>      and     al, al
459                  <1>      ;jnz short sysrel2 ; 0FFh ; user mode
460 0000C50F 75E4      <1>      jnz     short sysrel4 ; 29/01/2017
461 0000C511 FEC8      <1>      dec     al
462 0000C513 A2[1D4F0100]   <1>      mov     [u.t_mode], al ; 0FFh ; not necessary !?
463                  <1> sysrel6:
464                  <1>      ; cpu will continue from the interrupted sytem call addr
465 0000C518 61      <1>      popad      ; edi, esi, ebp, esp, ebx, edx, ecx, eax
466 0000C519 83C410   <1>      add     esp, 16      ; pass segment registers: ds, es, fs, gs
467 0000C51C CF      <1>      iretd      ; eip, cs, eflags
468                  <1>
469                  <1> sysrel7:
470 0000C51D 0FB61D[FB4E0100] <1>      movzx   ebx, byte [u.uno] ; current process number
471 0000C524 66C1E302   <1>      shl     bx, 2
472                  <1>      ;cmp [ebx+p.tcb-4], eax ; 0 ; is there callback address ?
473                  <1>      ;jna short sysrel0
474                  <1>      ; yes, reset callback address then restore process registers
475                  <1>      ;mov [ebx+p.tcb-4], eax ; 0 ; reset
476 0000C528 8B83[024C0100] <1>      mov     eax, [ebx+p.upage-4] ; UPAGE address
477 0000C52E FA      <1>      cli      ; disable interrupts till 'iretd'
478 0000C52F E979210000 <1>      jmp     rswap ; restore process 'u' structure
479                  <1>
480                  <1> badsys:
481                  <1>      ; 25/12/2016
482                  <1>      ; 18/04/2016 (TRDOS 386 = TRDOS v2.0)
483                  <1>      ; 17/04/2011 (TRDOS v1.0, 'IFC.ASM')
484                  <1>      ; 03/02/2011 ('trdos_ifc_routine')
485                  <1>      ;
486                  <1>      ; 16/04/2015 (Retro UNIX 386 v1, 'badsys')
487                  <1>      ; (EIP, EAX values will be shown on screen with error message)
488                  <1>      ; (EIP = 'CD 40h' instruction address -INT 40h-)
489                  <1>      ; (EAX = Function number)
490                  <1>      ;
491 0000C534 FE05[FA4E0100] <1>      inc     byte [u.bsys]
492                  <1>      ;
493 0000C53A 8B1D[A44E0100] <1>      mov     ebx, [u.sp] ; esp at the beginning of 'sysent'
494 0000C540 8B03      <1>      mov     eax, [ebx] ; EIP (return address, not 'INT 30h' address)
495 0000C542 83E802   <1>      sub     eax, 2 ; CDh, ##h
496 0000C545 E8BF6DFFFF   <1>      call    dwordtohex
497 0000C54A 8915[B5010100] <1>      mov     [eip_str], edx
498 0000C550 A3[B9010100]   <1>      mov     [eip_str+4], eax
499 0000C555 A1[AC4E0100]   <1>      mov     eax, [u.r0]
500 0000C55A E8AA6DFFFF   <1>      call    dwordtohex
501 0000C55F 8915[A4010100] <1>      mov     [eax_str], edx
502 0000C565 A3[A8010100]   <1>      mov     [eax_str+4], eax
503                  <1>
504 0000C56A 66C705[99010100]34- <1>      mov     word [int_num_str], SYSCALL_INT_NUM ; 25/12/2016
504 0000C572 30      <1>
505                  <1>
506 0000C573 BE[6B010100] <1>      mov     esi, ifc_msg ; "invalid funtion call !" msg (trdosk9.s)
507 0000C578 E8409DFFFF   <1>      call    print_msg
508                  <1>
509 0000C57D EB17      <1>      jmp     sysexit
510                  <1>
511                  <1> intract: ; / interrupt action
512                  <1>      ; 14/10/2015
513                  <1>      ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
514                  <1>      ; 09/05/2013 - 07/12/2013 (Retro UNIX 8086 v1)
515                  <1>      ;
516                  <1>      ; Retro UNIX 8086 v1 modification !
517                  <1>      ; (Process/task switching and quit routine by using
518                  <1>      ; Retro UNIX 8086 v1 keyboard interrupt output.))
519                  <1>      ;
520                  <1>      ; input -> 'u.quit' (also value of 'u.intr' > 0)
521                  <1>      ; output -> If value of 'u.quit' = FFFFh ('ctrl+brk' sign)
522                  <1>      ; 'intract' will jump to 'sysexit'.
523                  <1>      ; Intract will return to the caller
524                  <1>      ; if value of 'u.quit' <> FFFFh.
525                  <1>      ; 14/10/2015
526 0000C57F FB      <1>      sti
527                  <1>      ; 07/12/2013
528 0000C580 66FF05[F44E0100] <1>      inc     word [u.quit]
529 0000C587 7408      <1>      jz      short intrct0 ; FFFFh -> 0
530 0000C589 66FF0D[F44E0100] <1>      dec     word [u.quit]
531                  <1>      ; 16/04/2015
532 0000C590 C3      <1>      retn
533                  <1> intrct0:
534 0000C591 58      <1>      pop     eax ; call intract -> retn
535                  <1>      ;
536 0000C592 31C0      <1>      xor     eax, eax
537 0000C594 FEC0      <1>      inc     al ; mov ax, 1
538                  <1> ;;;
539                  <1>      ; UNIX v1 original 'intract' routine...
540                  <1>      ; / interrupt action
541                  <1>      ;cmp *(sp),$rti / are you in a clock interrupt?
542                  <1>      ; bne lf / no, lf
543                  <1>      ; cmp (sp)+,(sp)+ / pop clock pointer
544                  <1>      ; 1: / now in user area
545                  <1>      ; mov r1,-(sp) / save r1
546                  <1>      ; mov u.ttyp,r1
547                  <1>      ; / pointer to tty buffer in control-to r1
548                  <1>      ; cmpb 6(r1),$177
549                  <1>      ; / is the interrupt char equal to "del"

```

```

550          <1>          ; beq 1f / yes, 1f
551          <1>          ; clrb 6(r1)
552          <1>          ; / no, clear the byte
553          <1>          ; / (must be a quit character)
554          <1>          ; mov (sp)+,r1 / restore r1
555          <1>          ; clr u.quit / clear quit flag
556          <1>          ; bis $20,2(sp)
557          <1>          ; / set trace for quit (sets t bit of
558          <1>          ; / ps-trace trap)
559          <1>          ; rti ; / return from interrupt
560          <1>          ; 1: / interrupt char = del
561          <1>          ; clrb 6(r1) / clear the interrupt byte
562          <1>          ; / in the buffer
563          <1>          ; mov (sp)+,r1 / restore r1
564          <1>          ; cmp u.intr,$core / should control be
565          <1>          ; / transferred to loc core?
566          <1>          ; blo 1f
567          <1>          ; jmp *u.intr / user to do rti yes,
568          <1>          ; / transfer to loc core
569          <1>          ; 1:
570          <1>          ; sys 1 / exit
571          <1>
572          <1> sysexit: ; <terminate process>
573          <1>          ; 28/02/2017
574          <1>          ; 26/02/2017
575          <1>          ; 02/01/2017, 23/01/2017
576          <1>          ; 06/06/2016, 10/06/2016
577          <1>          ; 19/05/2016, 23/05/2016
578          <1>          ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
579          <1>          ; 16/04/2015 - 01/09/2015 (Retro UNIX 386 v1)
580          <1>          ; 19/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
581          <1>          ;
582          <1>          ; 'sysexit' terminates a process. First each file that
583          <1>          ; the process has opened is closed by 'flose'. The process
584          <1>          ; status is then set to unused. The 'p.pid' table is then
585          <1>          ; searched to find children of the dying process. If any of
586          <1>          ; children are zombies (died by not waited for), they are
587          <1>          ; set free. The 'p.pid' table is then searched to find the
588          <1>          ; dying process's parent. When the parent is found, it is
589          <1>          ; checked to see if it is free or it is a zombie. If it is
590          <1>          ; one of these, the dying process just dies. If it is waiting
591          <1>          ; for a child process to die, it notified that it doesn't
592          <1>          ; have to wait anymore by setting it's status from 2 to 1
593          <1>          ; (waiting to active). It is awakened and put on runq by
594          <1>          ; 'putlu'. The dying process enters a zombie state in which
595          <1>          ; it will never be run again but stays around until a 'wait'
596          <1>          ; is completed by it's parent process. If the parent is not
597          <1>          ; found, process just dies. This means 'swap' is called with
598          <1>          ; 'u.uno=0'. What this does is the 'wswap' is not called
599          <1>          ; to write out the process and 'rswap' reads the new process
600          <1>          ; over the one that dies..i.e., the dying process is
601          <1>          ; overwritten and destroyed.
602          <1>          ;
603          <1>          ; Calling sequence:
604          <1>          ;     sysexit or conditional branch.
605          <1>          ; Arguments:
606          <1>          ;     -
607          <1>          ; .....
608          <1>          ;
609          <1>          ; Retro UNIX 8086 v1 modification:
610          <1>          ;     System call number (=1) is in EAX register.
611          <1>          ;
612          <1>          ;     Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
613          <1>          ;     registers depending of function details.
614          <1>          ;
615          <1>          ; ('swap' procedure is mostly different than original UNIX v1.)
616          <1>          ;
617          <1>          ; / terminate process
618          <1>          ; AX = 1
619          0000C596 6648          <1>          dec     ax ; 0
620          0000C598 66A3[F24E0100] <1>          mov     [u.intr], ax ; 0
621          <1>          ; clr u.intr / clear interrupt control word
622          <1>          ; clr r1 / clear r1
623          <1>          sysexit_0:
624          <1>          ; 23/01/2017
625          <1>          ; 02/01/2017
626          <1>          ; 10/06/2016
627          <1>          ; 06/06/2016
628          <1>          ; 23/05/2016
629          <1>          ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
630          <1>          ; Check and stop/clear timer event(s) of this (dying) process
631          <1>          ; if there is.
632          <1>
633          <1>          ; 02/01/2017
634          0000C59E FA          <1>          cli     ; disable interrupts
635          <1>          ; 23/01/2017 - reset timer frequency (to 18.2Hz)
636          0000C59F B036          <1>          mov     al, 00110110b ; 36h
637          0000C5A1 E643          <1>          out     43h, al
638          0000C5A3 28C0          <1>          sub     al, al ; 0
639          0000C5A5 E640          <1>          out     40h, al ; LB
640          0000C5A7 E640          <1>          out     40h, al ; HB
641          <1>          ;
642          0000C5A9 0FB61D[FB4E0100] <1>          movzx  ebx, byte [u.uno]
643          <1>          ;mov     bl, [u.uno] ; process number of dying process
644          0000C5B0 3883[454C0100] <1>          cmp     byte [ebx+p.timer-1], al ; 0
645          0000C5B6 763B          <1>          jna     short sysexit_12 ; no timer events for this process
646          0000C5B8 8883[454C0100] <1>          mov     byte [ebx+p.timer-1], al ; 0 ; reset
647          0000C5BE A0[B7450100] <1>          mov     al, [timer_events]
648          <1>          ;or     al, al
649          <1>          ;jz     short sysexit_12 ; no timer events
650          0000C5C3 88C1          <1>          mov     cl, al
651          <1>          ;cli     ; disable interrupts

```

```

652 0000C5C5 B410      <1>      mov     ah, 16 ; number of available timer events
653 0000C5C7 BE[A84F0100] <1>      mov     esi, timer_set ; beginning address of timer events
654                                <1> sysexit_7:
655 0000C5CC 8A06      <1>      mov     al, [esi] ; process number (of timer event)
656 0000C5CE 38D8      <1>      cmp     al, bl ; process number comparison
657 0000C5D0 7411      <1>      je      short sysexit_10
658 0000C5D2 20C0      <1>      and     al, al
659 0000C5D4 7404      <1>      jz      short sysexit_9
660                                <1> sysexit_8:
661 0000C5D6 FEC9      <1>      dec     cl
662 0000C5D8 7416      <1>      jz      short sysexit_11
663                                <1> sysexit_9:
664 0000C5DA FECC      <1>      dec     ah
665 0000C5DC 7415      <1>      jz      short sysexit_12
666 0000C5DE 83C610    <1>      add     esi, 16
667 0000C5E1 EBE9      <1>      jmp     short sysexit_7
668                                <1>
669                                <1> sysexit_10:
670                                <1>      ;mov     byte [esi], 0
671 0000C5E3 66C7060000    <1>      mov     word [esi], 0
672                                <1>      ;mov     dword [esi+12], 0
673                                <1>      ;
674 0000C5E8 FE0D[B7450100] <1>      dec     byte [timer_events] ; 02/01/2017
675                                <1>      ;
676 0000C5EE EBE6      <1>      jmp     short sysexit_8
677                                <1>
678                                <1> sysexit_11:
679 0000C5F0 6629C0    <1>      sub     ax, ax ; 0 ; 26/02/2017
680                                <1> sysexit_12:
681                                <1>      ; 26/02/2017 (Unlink IRQ callbacks belong to the user)
682 0000C5F3 803D[1E4F0100]00 <1>      cmp     byte [u.irqc], 0 ; Count of IRQ callbacks
683 0000C5FA 7E2E      <1>      jng     short sysexit_16 ; zero or invalid
684                                <1>      ; 28/02/2017
685                                <1>      ; clear IRQ callback flags (for 'sysrele' and 'sysret')
686 0000C5FC A2[1F4F0100]    <1>      mov     [u.irqwait], al ; 0 ; force to clear waiting flag
687 0000C601 A2[204F0100]    <1>      mov     [u.r_lock], al ; 0 ; force to clear busy flag
688 0000C606 BE[F84A0100]    <1>      mov     esi, IRQ.owner
689                                <1> sysexit_13:
690 0000C60B AC          <1>      lodsb
691 0000C60C 3A05[FB4E0100]    <1>      cmp     al, [u.uno] ; owner = current user ?
692 0000C612 750C      <1>      jne     short sysexit_14
693 0000C614 C646FF00    <1>      mov     byte [esi-1], 0 ; owner = 0 : Free
694 0000C618 FE0D[1E4F0100]    <1>      dec     byte [u.irqc]
695 0000C61E 7408      <1>      jz      short sysexit_15
696                                <1> sysexit_14:
697 0000C620 81FE[004B0100]    <1>      cmp     esi, IRQ.owner + 8 ; the last IRQ index number ?
698 0000C626 76E3      <1>      jna     short sysexit_13 ; no
699                                <1> sysexit_15:
700 0000C628 30C0      <1>      xor     al, al ; 0
701                                <1> sysexit_16: ; 2:
702 0000C62A FB          <1>      sti     ; enable interrupts
703                                <1>      ;
704                                <1>      ; AX = 0
705                                <1> sysexit_1: ; 1:
706                                <1>      ; AX = File descriptor
707                                <1>      ; / r1 has file descriptor (index to u.fp list)
708                                <1>      ; / Search the whole list
709 0000C62B E8D3130000    <1>      call    fclose
710                                <1>      ; jsr r0,fclose / close all files the process opened
711                                <1>      ;; ignore error return
712                                <1>      ; br .+2 / ignore error return
713                                <1>      ;inc     ax
714 0000C630 FEC0      <1>      inc     al
715                                <1>      ; inc r1 / increment file descriptor
716                                <1>      ;cmp     ax, 10
717 0000C632 3C0A      <1>      cmp     al, 10
718                                <1>      ; cmp r1,$10. / end of u.fp list?
719 0000C634 72F5      <1>      jb      short sysexit_1
720                                <1>      ; blt 1b / no, go back
721                                <1>      ;movzx   ebx, byte [u.uno]
722 0000C636 8A1D[FB4E0100]    <1>      mov     bl, [u.uno] ; 02/01/2017
723                                <1>      ; movb u.uno,r1 / yes, move dying process's number to r1
724 0000C63C 88A3[F54B0100]    <1>      mov     [ebx+p.stat-1], ah ; 0, SFREE
725                                <1>      ; clrb p.stat-1(r1) / free the process
726                                <1>      ;shl     bx, 1
727 0000C642 D0E3      <1>      shl     bl, 1
728                                <1>      ; asl r1 / use r1 for index into the below tables
729 0000C644 668B8B[644B0100]    <1>      mov     cx, [ebx+p.pid-2]
730                                <1>      ; mov p.pid-2(r1),r3 / move dying process's name to r3
731 0000C64B 668B93[844B0100]    <1>      mov     dx, [ebx+p.ppid-2]
732                                <1>      ; mov p.ppid-2(r1),r4 / move its parents name to r4
733                                <1>      ; xor     bx, bx ; 0
734 0000C652 30DB      <1>      xor     bl, bl ; 0
735                                <1>      ; clr     r2
736 0000C654 31F6      <1>      xor     esi, esi ; 0
737                                <1>      ; clr     r5 / initialize reg
738                                <1> sysexit_2: ; 1:
739                                <1>      ; / find children of this dying process,
740                                <1>      ; / if they are zombies, free them
741                                <1>      ;add     bx, 2
742 0000C656 80C302      <1>      add     bl, 2
743                                <1>      ; add $2,r2 / search parent process table
744                                <1>      ; / for dying process's name
745 0000C659 66398B[844B0100]    <1>      cmp     [ebx+p.ppid-2], cx
746                                <1>      ; cmp p.ppid-2(r2),r3 / found it?
747 0000C660 7513      <1>      jne     short sysexit_4
748                                <1>      ; bne 3f / no
749                                <1>      ;shr     bx, 1
750 0000C662 D0EB      <1>      shr     bl, 1
751                                <1>      ; asr r2 / yes, it is a parent
752 0000C664 80BB[F54B0100]03 <1>      cmp     byte [ebx+p.stat-1], 3 ; SZOMB
753                                <1>      ; cmpb p.stat-1(r2),$3 / is the child of this

```

```

754                                     <1>                                     ; / dying process a zombie
755 0000C66B 7506                       <1>         jne      short sysexit_3
756                                     <1>         ; bne 2f / no
757 0000C66D 88A3[F54B0100]             <1>         mov      [ebx+p.stat-1], ah ; 0, SFREE
758                                     <1>         ; clrb p.stat-1(r2) / yes, free the child process
759                                     <1> sysexit_3: ; 2:
760                                     <1>         ;shr     bx, 1
761 0000C673 D0E3                       <1>         shl      bl, 1
762                                     <1>         ; asl r2
763                                     <1> sysexit_4: ; 3:
764                                     <1>         ; / search the process name table
765                                     <1>         ; / for the dying process's parent
766 0000C675 663993[644B0100]           <1>         cmp      [ebx+p.pid-2], dx
767                                     <1>         ; cmp p.pid-2(r2),r4 / found it?
768 0000C67C 7502                       <1>         jne      short sysexit_5
769                                     <1>         ; bne 3f / no
770 0000C67E 89DE                       <1>         mov      esi, ebx
771                                     <1>         ; mov r2,r5 / yes, put index to p.pid table (parents
772                                     <1>         ; / process # x2) in r5
773                                     <1> sysexit_5: ; 3:
774                                     <1>         ;cmp     bx, nproc + nproc
775 0000C680 80FB20                     <1>         cmp      bl, nproc + nproc
776                                     <1>         ; cmp r2,$nproc+nproc / has whole table been searched?
777 0000C683 72D1                       <1>         jnb     short sysexit_2
778                                     <1>         ; blt 1b / no, go back
779                                     <1>         ; mov r5,r1 / yes, r1 now has parents process # x2
780 0000C685 21F6                       <1>         and      esi, esi ; r5=r1
781 0000C687 7436                       <1>         jz       short sysexit_6
782                                     <1>         ; beq 2f / no parent has been found.
783                                     <1>         ; / The process just dies
784 0000C689 66D1EE                     <1>         shr      si, 1
785                                     <1>         ; asr r1 / set up index to p.stat
786 0000C68C 8A86[F54B0100]             <1>         mov      al, [esi+p.stat-1]
787                                     <1>         ; movb p.stat-1(r1),r2 / move status of parent to r2
788 0000C692 20C0                       <1>         and      al, al
789 0000C694 7429                       <1>         jz       short sysexit_6
790                                     <1>         ; beq 2f / if its been freed, 2f
791 0000C696 3C03                       <1>         cmp      al, 3
792                                     <1>         ; cmp r2,$3 / is parent a zombie?
793 0000C698 7425                       <1>         je       short sysexit_6
794                                     <1>         ; beq 2f / yes, 2f
795                                     <1>         ; BH = 0
796 0000C69A 8A1D[FB4E0100]             <1>         mov      bl, [u.uno]
797                                     <1>         ; movb u.uno,r3 / move dying process's number to r3
798 0000C6A0 C683[F54B0100]03           <1>         mov      byte [ebx+p.stat-1], 3 ; SZOMB
799                                     <1>         ; movb $3,p.stat-1(r3) / make the process a zombie
800 0000C6A7 3C01                       <1>         cmp      al, 1 ; SRUN
801 0000C6A9 7414                       <1>         je       short sysexit_6
802                                     <1>         ;cmp     al, 2
803                                     <1>         ; cmp r2,$2 / is the parent waiting for
804                                     <1>         ; / this child to die
805                                     <1>         ;jne     short sysexit_6
806                                     <1>         ; bne 2f / yes, notify parent not to wait any more
807                                     <1>         ; p.stat = 2 --> waiting
808                                     <1>         ; p.stat = 4 --> sleeping
809 0000C6AB C686[F54B0100]01           <1>         mov      byte [esi+p.stat-1], 1 ; SRUN
810                                     <1>         ;dec     byte [esi+p.stat-1]
811                                     <1>         ; decbp.stat-1(r1) / awaken it by putting it (parent)
812 0000C6B2 6689F0                     <1>         mov      ax, si ; r1 (process number in AL)
813                                     <1>         ;
814                                     <1>         ;mov     ebx, runq + 4
815                                     <1>         ; mov $runq+4,r2 / on the runq
816 0000C6B5 BB[9A4E0100]               <1>         mov      ebx, runq+2 ; normal run queue ; 02/01/2017
817 0000C6BA E826200000                 <1>         call     putlu
818                                     <1>         ; jsr r0, putlu
819                                     <1> sysexit_6:
820                                     <1>         ; / the process dies
821 0000C6BF C605[FB4E0100]00           <1>         mov      byte [u.uno], 0
822                                     <1>         ; clrb u.uno / put zero as the process number,
823                                     <1>         ; / so "swap" will
824 0000C6C6 E81C1F0000                 <1>         call     swap
825                                     <1>         ; jsr r0,swap / overwrite process with another process
826                                     <1>
827                                     <1> hlt_sys:
828                                     <1>         ;sti
829                                     <1> hlts0:
830 0000C6CB F4                         <1>         hlt
831 0000C6CC EBFD                       <1>         jmp      short hlts0
832                                     <1>         ; 0 / and thereby kill it; halt?
833                                     <1>
834                                     <1> syswait: ; < wait for a processs to die >
835                                     <1>         ; 17/09/2015
836                                     <1>         ; 02/09/2015
837                                     <1>         ; 01/09/2015
838                                     <1>         ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
839                                     <1>         ; 24/05/2013 - 05/02/2014 (Retro UNIX 8086 v1)
840                                     <1>         ;
841                                     <1>         ; 'syswait' waits for a process die.
842                                     <1>         ; It works in following way:
843                                     <1>         ; 1) From the parent process number, the parent's
844                                     <1>         ; process name is found. The p.ppid table of parent
845                                     <1>         ; names is then searched for this process name.
846                                     <1>         ; If a match occurs, r2 contains child's process
847                                     <1>         ; number. The child status is checked to see if it is
848                                     <1>         ; a zombie, i.e; dead but not waited for (p.stat=3)
849                                     <1>         ; If it is, the child process is freed and it's name
850                                     <1>         ; is put in (u.r0). A return is then made via 'sysret'.
851                                     <1>         ; If the child is not a zombie, nothing happens and
852                                     <1>         ; the search goes on through the p.ppid table until
853                                     <1>         ; all processes are checked or a zombie is found.
854                                     <1>         ; 2) If no zombies are found, a check is made to see if
855                                     <1>         ; there are any children at all. If there are none,

```



```

856      <1>      ;      an error return is made. If there are, the parent's
857      <1>      ;      status is set to 2 (waiting for child to die),
858      <1>      ;      the parent is swapped out, and a branch to 'syswait'
859      <1>      ;      is made to wait on the next process.
860      <1>      ;
861      <1>      ; Calling sequence:
862      <1>      ;      ?
863      <1>      ; Arguments:
864      <1>      ;      -
865      <1>      ; Inputs: -
866      <1>      ; Outputs: if zombie found, it's name put in u.r0.
867      <1>      ; .....
868      <1>      ;
869      <1>
870      <1> ; / wait for a process to die
871      <1>
872      <1> syswait_0:
873 0000C6CE 0FB61D[FB4E0100] <1>      movzx ebx, byte [u.uno] ; 01/09/2015
874      <1>      ; movb u.uno,r1 / put parents process number in r1
875 0000C6D5 D0E3      <1>      shl bl, 1
876      <1>      ;shl bx, 1
877      <1>      ; asl r1 / x2 to get index into p.pid table
878 0000C6D7 668B83[644B0100] <1>      mov ax, [ebx+p.pid-2]
879      <1>      ; mov p.pid-2(r1),r1 / get the name of this process
880 0000C6DE 31F6      <1>      xor esi, esi
881      <1>      ; clr r2
882 0000C6E0 31C9      <1>      xor ecx, ecx ; 30/10/2013
883      <1>      ;xor cl, cl
884      <1>      ; clr r3 / initialize reg 3
885      <1> syswait_1: ; 1:
886 0000C6E2 6683C602 <1>      add si, 2
887      <1>      ; add $2,r2 / use r2 for index into p.ppid table
888      <1>      ; / search table of parent processes
889      <1>      ; / for this process name
890 0000C6E6 663B86[844B0100] <1>      cmp ax, [esi+p.ppid-2]
891      <1>      ; cmp p.ppid-2(r2),r1 / r2 will contain the childs
892      <1>      ; / process number
893 0000C6ED 7535      <1>      jne short syswait_3
894      <1>      ;bne 3f / branch if no match of parent process name
895      <1>      ;inc cx
896 0000C6EF FEC1      <1>      inc cl
897      <1>      ;inc r3 / yes, a match, r3 indicates number of children
898 0000C6F1 66D1EE <1>      shr si, 1
899      <1>      ; asr r2 / r2/2 to get index to p.stat table
900      <1>      ; The possible states ('p.stat' values) of a process are:
901      <1>      ;      0 = free or unused
902      <1>      ;      1 = active
903      <1>      ;      2 = waiting for a child process to die
904      <1>      ;      3 = terminated, but not yet waited for (zombie).
905 0000C6F4 80BE[F54B0100]03 <1>      cmp byte [esi+p.stat-1], 3 ; SZOMB, 05/02/2014
906      <1>      ; cmpb p.stat-1(r2),$3 / is the child process a zombie?
907 0000C6FB 7524      <1>      jne short syswait_2
908      <1>      ; bne 2f / no, skip it
909 0000C6FD 88BE[F54B0100] <1>      mov [esi+p.stat-1], bh ; 0
910      <1>      ; clrb p.stat-1(r2) / yes, free it
911 0000C703 66D1E6 <1>      shl si, 1
912      <1>      ; asl r2 / r2x2 to get index into p.ppid table
913 0000C706 0FB786[644B0100] <1>      movzx eax, word [esi+p.pid-2]
914 0000C70D A3[AC4E0100] <1>      mov [u.r0], eax
915      <1>      ; mov p.pid-2(r2),*u.r0
916      <1>      ; / put childs process name in (u.r0)
917      <1>      ;
918      <1>      ; Retro UNIX 386 v1 modification ! (17/09/2015)
919      <1>      ;
920      <1>      ; Parent process ID -p.ppid- field (of the child process)
921      <1>      ; must be cleared in order to prevent infinitive 'syswait'
922      <1>      ; system call loop from the application/program if it calls
923      <1>      ; 'syswait' again (mistakenly) while there is not a zombie
924      <1>      ; or running child process to wait. ('forktest.s', 17/09/2015)
925      <1>      ;
926      <1>      ; Note: syswait will return with error if there is not a
927      <1>      ;      zombie or running process to wait.
928      <1>      ;
929 0000C712 6629C0 <1>      sub ax, ax
930 0000C715 668986[844B0100] <1>      mov [esi+p.ppid-2], ax ; 0 ; 17/09/2015
931 0000C71C E9F0FCFFFF <1>      jmp sysret0 ; ax = 0
932      <1>      ;
933      <1>      ;jmp sysret
934      <1>      ; br sysret1 / return cause child is dead
935      <1> syswait_2: ; 2:
936 0000C721 66D1E6 <1>      shl si, 1
937      <1>      ; asl r2 / r2x2 to get index into p.ppid table
938      <1> syswait_3: ; 3:
939 0000C724 6683FE20 <1>      cmp si, nproc+nproc
940      <1>      ; cmp r2,$nproc+nproc / have all processes been checked?
941 0000C728 72B8      <1>      jb short syswait_1
942      <1>      ; blt 1b / no, continue search
943      <1>      ;and cx, cx
944 0000C72A 20C9      <1>      and cl, cl
945      <1>      ; tst r3 / one gets here if there are no children
946      <1>      ; / or children that are still active
947      <1>      ; 30/10/2013
948 0000C72C 750B      <1>      jnz short syswait_4
949      <1>      ;jz error
950      <1>      ; beq error1 / there are no children, error
951 0000C72E 890D[AC4E0100] <1>      mov [u.r0], ecx ; 0
952 0000C734 E9B6FCFFFF <1>      jmp error
953      <1> syswait_4:
954 0000C739 8A1D[FB4E0100] <1>      mov bl, [u.uno]
955      <1>      ; movb u.uno,r1 / there are children so put
956      <1>      ; / parent process number in r1
957 0000C73F FE83[F54B0100] <1>      inc byte [ebx+p.stat-1] ; 2, SWAIT, 05/02/2014

```

```

958          <1>          ; incb p.stat-1(r1) / it is waiting for
959          <1>          ; / other children to die
960          <1>          ; 04/11/2013
961 0000C745 E89D1E0000 <1>          call    swap
962          <1>          ; jsr r0,swap / swap it out, because it's waiting
963 0000C74A EB82      <1>          jmp     syswait_0
964          <1>          ; br syswait / wait on next process
965          <1>
966          <1> sysfork: ; < create a new process >
967          <1>          ; 02/01/2017 (TRDOS 386 modification)
968          <1>          ; 04/09/2015, 18/05/2015
969          <1>          ; 28/08/2015, 01/09/2015, 02/09/2015
970          <1>          ; 09/05/2015, 10/05/2015, 14/05/2015
971          <1>          ; 06/05/2015 (Retro UNIX 386 v1 - Beginning)
972          <1>          ; 24/05/2013 - 14/02/2014 (Retro UNIX 8086 v1)
973          <1>          ;
974          <1>          ; 'sysfork' creates a new process. This process is referred
975          <1>          ; to as the child process. This new process core image is
976          <1>          ; a copy of that of the caller of 'sysfork'. The only
977          <1>          ; distinction is the return location and the fact that (u.r0)
978          <1>          ; in the old process (parent) contains the process id (p.pid)
979          <1>          ; of the new process (child). This id is used by 'syswait'.
980          <1>          ; 'sysfork' works in the following manner:
981          <1>          ; 1) The process status table (p.stat) is searched to find
982          <1>          ; a process number that is unused. If none are found
983          <1>          ; an error occurs.
984          <1>          ; 2) when one is found, it becomes the child process number
985          <1>          ; and it's status (p.stat) is set to active.
986          <1>          ; 3) If the parent had a control tty, the interrupt
987          <1>          ; character in that tty buffer is cleared.
988          <1>          ; 4) The child process is put on the lowest priority run
989          <1>          ; queue via 'putlu'.
990          <1>          ; 5) A new process name is gotten from 'mpid' (actually
991          <1>          ; it is a unique number) and is put in the child's unique
992          <1>          ; identifier; process id (p.pid).
993          <1>          ; 6) The process name of the parent is then obtained and
994          <1>          ; placed in the unique identifier of the parent process
995          <1>          ; name is then put in 'u.r0'.
996          <1>          ; 7) The child process is then written out on disk by
997          <1>          ; 'wswap', i.e., the parent process is copied onto disk
998          <1>          ; and the child is born. (The child process is written
999          <1>          ; out on disk/drum with 'u.uno' being the child process
1000         <1>          ; number.)
1001         <1>          ; 8) The parent process number is then restored to 'u.uno'.
1002         <1>          ; 9) The child process name is put in 'u.r0'.
1003         <1>          ; 10) The pc on the stack sp + 18 is incremented by 2 to
1004         <1>          ; create the return address for the parent process.
1005         <1>          ; 11) The 'u.fp' list is then searched to see what files
1006         <1>          ; the parent has opened. For each file the parent has
1007         <1>          ; opened, the corresponding 'fsp' entry must be updated
1008         <1>          ; to indicate that the child process also has opened
1009         <1>          ; the file. A branch to 'sysret' is then made.

1010         <1>          ;
1011         <1>          ; Calling sequence:
1012         <1>          ; from shell ?
1013         <1>          ; Arguments:
1014         <1>          ; -
1015         <1>          ; Inputs: -
1016         <1>          ; Outputs: *u.r0 - child process name
1017         <1>          ; .....
1018         <1>          ;
1019         <1>          ; Retro UNIX 8086 v1 modification:
1020         <1>          ; AX = r0 = PID (>0) (at the return of 'sysfork')
1021         <1>          ; = process id of child a parent process returns
1022         <1>          ; = process id of parent when a child process returns
1023         <1>          ;
1024         <1>          ; In original UNIX v1, sysfork is called and returns as
1025         <1>          ; in following manner: (with an example: c library, fork)
1026         <1>          ;
1027         <1>          ; 1:
1028         <1>          ; sys fork
1029         <1>          ; br 1f / child process returns here
1030         <1>          ; bes 2f / parent process returns here
1031         <1>          ; / pid of new process in r0
1032         <1>          ; rts pc
1033         <1>          ; 2: / parent process conditionally branches here
1034         <1>          ; mov $-1,r0 / pid = -1 means error return
1035         <1>          ; rts pc
1036         <1>          ;
1037         <1>          ; 1: / child process branches here
1038         <1>          ; clr r0 / pid = 0 in child process
1039         <1>          ; rts pc
1040         <1>          ;
1041         <1>          ; In UNIX v7x86 (386) by Robert Nordier (1999)
1042         <1>          ; // pid = fork();
1043         <1>          ; //
1044         <1>          ; // pid == 0 in child process;
1045         <1>          ; // pid == -1 means error return
1046         <1>          ; // in child,
1047         <1>          ; // parents id is in par_uid if needed
1048         <1>          ;
1049         <1>          ; _fork:
1050         <1>          ; mov $.fork,eax
1051         <1>          ; int $0x30
1052         <1>          ; jmp 1f
1053         <1>          ; jnc 2f
1054         <1>          ; jmp cerror
1055         <1>          ;
1056         <1>          ; 1:
1057         <1>          ; mov eax,_par_uid
1058         <1>          ; xor eax,eax

```

```

1059 <1> ; ret
1060 <1> ;
1061 <1> ; In Retro UNIX 8086 v1,
1062 <1> ; 'sysfork' returns in following manner:
1063 <1> ;
1064 <1> ; mov ax, sys_fork
1065 <1> ; mov bx, offset @f ; routine for child
1066 <1> ; int 20h
1067 <1> ; jc error
1068 <1> ;
1069 <1> ; ; Routine for parent process here (just after 'jc')
1070 <1> ; mov word ptr [pid_of_child], ax
1071 <1> ; jmp next_routine_for_parent
1072 <1> ;
1073 <1> ; @@: ; routine for child process here
1074 <1> ; ....
1075 <1> ; NOTE: 'sysfork' returns to specified offset
1076 <1> ; for child process by using BX input.
1077 <1> ; (at first, parent process will return then
1078 <1> ; child process will return -after swapped in-
1079 <1> ; 'syswait' is needed in parent process
1080 <1> ; if return from child process will be waited for.)
1081 <1> ;
1082 <1> ;
1083 <1> ; / create a new process
1084 <1> ; EBX = return address for child process
1085 <1> ; ; (Retro UNIX 8086 v1 modification !)
1086 0000C74C 31F6 <1> xor esi, esi
1087 <1> ; clr r1
1088 <1> sysfork_1: ; 1: / search p.stat table for unused process number
1089 0000C74E 46 <1> inc esi
1090 <1> ; inc r1
1091 0000C74F 80BE[F54B0100]00 <1> cmp byte [esi+p.stat-1], 0 ; SFREE, 05/02/2014
1092 <1> ; tstb p.stat-1(r1) / is process active, unused, dead
1093 0000C756 760B <1> jna short sysfork_2
1094 <1> ; beq 1f / it's unused so branch
1095 0000C758 6683FE10 <1> cmp si, nproc
1096 <1> ; cmp r1,$nproc / all processes checked
1097 0000C75C 72F0 <1> jnb short sysfork_1
1098 <1> ; blt 1b / no, branch back
1099 <1> ;
1100 <1> ; Retro UNIX 8086 v1. modification:
1101 <1> ; Parent process returns from 'sysfork' to address
1102 <1> ; which is just after 'sysfork' system call in parent
1103 <1> ; process. Child process returns to address which is put
1104 <1> ; in BX register by parent process for 'sysfork'.
1105 <1> ;
1106 <1> ; add $2,18.(sp) / add 2 to pc when trap occurred, points
1107 <1> ; / to old process return
1108 <1> ; br error1 / no room for a new process
1109 0000C75E E98CFCFFFF <1> jmp error
1110 <1> sysfork_2: ; 1:
1111 0000C763 E81184FFFF <1> call allocate_page
1112 0000C768 0F8281FCFFFF <1> jc error
1113 0000C76E 50 <1> push eax ; UPAGE (user structure page) address
1114 <1> ; Retro UNIX 386 v1 modification!
1115 0000C76F E81486FFFF <1> call duplicate_page_dir
1116 <1> ; EAX = New page directory
1117 0000C774 730B <1> jnc short sysfork_3
1118 0000C776 58 <1> pop eax ; UPAGE (user structure page) address
1119 0000C777 E8DB85FFFF <1> call deallocate_page
1120 0000C77C E96EFCFFFF <1> jmp error
1121 <1> sysfork_3:
1122 <1> ; Retro UNIX 386 v1 modification !
1123 0000C781 56 <1> push esi
1124 0000C782 E8EE1E0000 <1> call wswap ; save current user (u) structure, user registers
1125 <1> ; and interrupt return components (for IRET)
1126 0000C787 8705[004F0100] <1> xchg eax, [u.pgdir] ; page directory of the child process
1127 0000C78D A3[044F0100] <1> mov [u.ppgdir], eax ; page directory of the parent process
1128 0000C792 5E <1> pop esi
1129 0000C793 58 <1> pop eax ; UPAGE (user structure page) address
1130 <1> ; [u.usp] = esp
1131 0000C794 89F7 <1> mov edi, esi
1132 0000C796 66C1E702 <1> shl di, 2
1133 0000C79A 8987[024C0100] <1> mov [edi+p.upage-4], eax ; memory page for 'user' struct
1134 0000C7A0 A3[FC4E0100] <1> mov [u.upage], eax ; memory page for 'user' struct (child)
1135 <1> ; 28/08/2015
1136 0000C7A5 0FB605[FB4E0100] <1> movzx eax, byte [u.uno] ; parent process number
1137 <1> ; movb u.uno,-(sp) / save parent process number
1138 0000C7AC 89C7 <1> mov edi, eax
1139 0000C7AE 50 <1> pusheax ; **
1140 0000C7AF 8A87[C54B0100] <1> mov al, [edi+p.ttyc-1] ; console tty (parent)
1141 <1> ; 18/09/2015
1142 <1> ; mov [esi+p.ttyc-1], al ; set child's console tty
1143 <1> ; mov [esi+p.waitc-1], ah ; 0 ; reset child's wait channel
1144 0000C7B5 668986[C54B0100] <1> mov [esi+p.ttyc-1], ax ; al - set child's console tty
1145 <1> ; ah - reset child's wait channel
1146 0000C7BC 89F0 <1> mov eax, esi
1147 0000C7BE A2[FB4E0100] <1> mov [u.uno], al ; child process number
1148 <1> ; movb r1,u.uno / set child process number to r1
1149 0000C7C3 FE86[F54B0100] <1> inc byte [esi+p.stat-1] ; 1, SRUN, 05/02/2014
1150 <1> ; incb p.stat-1(r1) / set p.stat entry for child
1151 <1> ; / process to active status
1152 <1> ; mov u.ttyp,r2 / put pointer to parent process'
1153 <1> ; / control tty buffer in r2
1154 <1> ; beq 2f / branch, if no such tty assigned
1155 <1> ; clrb 6(r2) / clear interrupt character in tty buffer
1156 <1> ; 2:
1157 0000C7C9 53 <1> push ebx ; * return address for the child process
1158 <1> ; * Retro UNIX 8086 v1 feature only !
1159 <1> ; (Retro UNIX 8086 v1 modification!)
1160 <1> ; mov $runq+4,r2

```

```

1161 0000C7CA BB[9A4E0100] <1> mov ebx, runq+2 ; normal run queue ; 02/01/2017
1162 0000C7CF E8111F0000 <1> call putlu
1163 <1> ; jsr r0,putlu / put child process on lowest priority
1164 <1> ; / run queue
1165 0000C7D4 66D1E6 <1> shl si, 1
1166 <1> ; asl r1 / multiply r1 by 2 to get index
1167 <1> ; / into p.pid table
1168 0000C7D7 66FF05[944E0100] <1> inc word [mpid]
1169 <1> ; inc mpid / increment m.pid; get a new process name
1170 0000C7DE 66A1[944E0100] <1> mov ax, [mpid]
1171 0000C7E4 668986[644B0100] <1> mov [esi+p.pid-2], ax
1172 <1> ;mov mpid,p.pid-2(r1) / put new process name
1173 <1> ; / in child process' name slot
1174 0000C7EB 5A <1> pop edx ; * return address for the child process
1175 <1> ; * Retro UNIX 8086 v1 feature only !
1176 0000C7EC 5B <1> pop ebx ; **
1177 <1> ;mov ebx, [esp] ; ** parent process number
1178 <1> ; movb (sp),r2 / put parent process number in r2
1179 0000C7ED 66D1E3 <1> shl bx, 1
1180 <1> ;asl r2 / multiply by 2 to get index into below tables
1181 <1> ;movzx eax, word [ebx+p.pid-2]
1182 0000C7F0 668B83[644B0100] <1> mov ax, [ebx+p.pid-2]
1183 <1> ; mov p.pid-2(r2),r2 / get process name of parent
1184 <1> ; / process
1185 0000C7F7 668986[844B0100] <1> mov [esi+p.ppid-2], ax
1186 <1> ; mov r2,p.ppid-2(r1) / put parent process name
1187 <1> ; / in parent process slot for child
1188 0000C7FE A3[AC4E0100] <1> mov [u.r0], eax
1189 <1> ; mov r2,*u.r0 / put parent process name on stack
1190 <1> ; / at location where r0 was saved
1191 0000C803 8B2D[A44E0100] <1> mov ebp, [u.sp] ; points to return address (EIP for IRET)
1192 0000C809 895500 <1> mov [ebp], edx ; *, CS:EIP -> EIP
1193 <1> ; * return address for the child process
1194 <1> ; mov $sysret1,-(sp) /
1195 <1> ; mov sp,u.usp / contents of sp at the time when
1196 <1> ; / user is swapped out
1197 <1> ; mov $sstack,sp / point sp to swapping stack space
1198 <1> ; 04/09/2015 - 01/09/2015
1199 <1> ; [u.usp] = esp
1200 0000C80C 68[0FC40000] <1> push sysret ; ***
1201 0000C811 8925[A84E0100] <1> mov [u.usp], esp ; points to 'sysret' address (***)
1202 <1> ; (for child process)
1203 0000C817 31C0 <1> xor eax, eax
1204 0000C819 66A3[DC4E0100] <1> mov [u.ttyp], ax ; 0
1205 <1> ;
1206 0000C81F E8511E0000 <1> call wswap ; Retro UNIX 8086 v1 modification !
1207 <1> ;jsr r0,wswap / put child process out on drum
1208 <1> ;jsr r0,unpack / unpack user stack
1209 <1> ;mov u.usp,sp / restore user stack pointer
1210 <1> ; tst (sp)+ / bump stack pointer
1211 <1> ; Retro UNIX 386 v1 modification !
1212 0000C824 58 <1> pop eax ; ***
1213 0000C825 66D1E3 <1> shl bx, 1
1214 0000C828 8B83[024C0100] <1> mov eax, [ebx+p.upage-4] ; UPAGE address ; 14/05/2015
1215 0000C82E E87A1E0000 <1> call rswap ; restore parent process 'u' structure,
1216 <1> ; registers and return address (for IRET)
1217 <1> ;movb (sp)+,u.uno / put parent process number in u.uno
1218 0000C833 0FB705[944E0100] <1> movzx eax, word [mpid]
1219 0000C83A A3[AC4E0100] <1> mov [u.r0], eax
1220 <1> ; mov mpid,*u.r0 / put child process name on stack
1221 <1> ; / where r0 was saved
1222 <1> ; add $2,18.(sp) / add 2 to pc on stack; gives parent
1223 <1> ; / process return
1224 <1> ;xor ebx, ebx
1225 0000C83F 31F6 <1> xor esi, esi
1226 <1> ;clr r1
1227 <1> sysfork_4: ; 1: / search u.fp list to find the files
1228 <1> ; / opened by the parent process
1229 <1> ; 01/09/2015
1230 <1> ;xor bh, bh
1231 <1> ;mov bl, [esi+u.fp]
1232 0000C841 8A86[B24E0100] <1> mov al, [esi+u.fp]
1233 <1> ; movb u.fp(r1),r2 / get an open file for this process
1234 <1> ;or bl, bl
1235 0000C847 08C0 <1> or al, al
1236 0000C849 740D <1> jz short sysfork_5
1237 <1> ; beq 2f / file has not been opened by parent,
1238 <1> ; / so branch
1239 0000C84B B40A <1> mov ah, 10 ; Retro UNIX 386 v1 fsp structure size = 10 bytes
1240 0000C84D F6E4 <1> mul ah
1241 <1> ;movzx ebx, ax
1242 0000C84F 6689C3 <1> mov bx, ax
1243 <1> ;shl bx, 3
1244 <1> ; asl r2 / multiply by 8
1245 <1> ; asl r2 / to get index into fsp table
1246 <1> ; asl r2
1247 0000C852 FE83[944C0100] <1> inc byte [ebx+fsp-2]
1248 <1> ; incb fsp-2(r2) / increment number of processes
1249 <1> ; / using file, because child will now be
1250 <1> ; / using this file
1251 <1> sysfork_5: ; 2:
1252 0000C858 46 <1> inc esi
1253 <1> ; inc r1 / get next open file
1254 0000C859 6683FE0A <1> cmp si, 10
1255 <1> ; cmp r1,$10. / 10. files is the maximum number which
1256 <1> ; / can be opened
1257 0000C85D 72E2 <1> jb short sysfork_4
1258 <1> ; blt 1b / check next entry
1259 0000C85F E9ABFBFFFF <1> jmp sysret
1260 <1> ; br sysret1
1261 <1>
1262 <1> syscreat: ; < create file >

```



```

1263 <1> ; 27/10/2016
1264 <1> ; 25/10/2016, 26/10/2016
1265 <1> ; 15/10/2016, 16/10/2016, 17/10/2016
1266 <1> ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)
1267 <1> ; -derived from INT_21H.ASM-
1268 <1> ; ("loc_INT21h_create_file")
1269 <1> ; 10/07/2011 (12/03/2011)
1270 <1> ; INT 21h Function AH = 3Ch
1271 <1> ; Create File
1272 <1> ; INPUT
1273 <1> ; CX = Attributes
1274 <1> ; DS:DX= Address of zero terminated path name
1275 <1> ;
1276 <1> ; 27/12/2015 (Retro UNIX 386 v1.1)
1277 <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1278 <1> ; 27/05/2013 (Retro UNIX 8086 v1)
1279 <1> ;
1280 <1> ; 'syscreat' called with two arguments; name and mode.
1281 <1> ; u.namep points to name of the file and mode is put
1282 <1> ; on the stack. 'namei' is called to get i-number of the file.
1283 <1> ; If the file already exists, it's mode and owner remain
1284 <1> ; unchanged, but it is truncated to zero length. If the file
1285 <1> ; did not exist, an i-node is created with the new mode via
1286 <1> ; 'maknod' whether or not the file already existed, it is
1287 <1> ; open for writing. The fsp table is then searched for a free
1288 <1> ; entry. When a free entry is found, proper data is placed
1289 <1> ; in it and the number of this entry is put in the u.fp list.
1290 <1> ; The index to the u.fp (also know as the file descriptor)
1291 <1> ; is put in the user's r0.
1292 <1> ;
1293 <1> ; Calling sequence:
1294 <1> ; syscreate; name; mode
1295 <1> ; Arguments:
1296 <1> ; name - name of the file to be created
1297 <1> ; mode - mode of the file to be created
1298 <1> ; Inputs: (arguments)
1299 <1> ; Outputs: *u.r0 - index to u.fp list
1300 <1> ; (the file descriptor of new file)
1301 <1> ; .....
1302 <1> ;
1303 <1> ; Retro UNIX 8086 v1 modification:
1304 <1> ; 'syscreate' system call has two arguments; so,
1305 <1> ; * 1st argument, name is pointed to by BX register
1306 <1> ; * 2nd argument, mode is in CX register
1307 <1> ;
1308 <1> ; AX register (will be restored via 'u.r0') will return
1309 <1> ; to the user with the file descriptor/number
1310 <1> ; (index to u.fp list).
1311 <1> ;
1312 <1> ;call arg2
1313 <1> ; * name - 'u.namep' points to address of file/path name
1314 <1> ; in the user's program segment ('u.segmt')
1315 <1> ; with offset in BX register (as sysopen argument 1).
1316 <1> ; * mode - sysopen argument 2 is in CX register
1317 <1> ; which is on top of stack.
1318 <1> ;
1319 <1> ; TRDOS 386 (10/10/2016)
1320 <1> ;
1321 <1> ; INPUT ->
1322 <1> ; CL = File Attributes
1323 <1> ; bit 0 (1) - Read only file (R)
1324 <1> ; bit 1 (1) - Hidden file (H)
1325 <1> ; bit 2 (1) - System file (R)
1326 <1> ; bit 3 (1) - Volume label/name (V)
1327 <1> ; bit 4 (1) - Subdirectory (D)
1328 <1> ; bit 5 (1) - File has been archived (A)
1329 <1> ; EBX = Pointer to filename (ASCIIZ) -path-
1330 <1> ;
1331 <1> ; OUTPUT ->
1332 <1> ; eax = File/Device Handle/Number (index) (AL)
1333 <1> ; cf = 1 -> Error code in AL
1334 <1> ;
1335 <1> ; Modified Registers: EAX (at the return of system call)
1336 <1> ;
1337 <1> ; Note: If the file is existing and it has not any one
1338 <1> ; of S,H,R,V,D attributes, it will be truncated
1339 <1> ; to zero length; otherwise, access error will be
1340 <1> ; returned.
1341 <1> ;
1342 <1> sysmkdir_0:
1343 0000C864 F6C108 <1> test cl, 08h ; Volume name
1344 0000C867 740A <1> jz short syscreat_0
1345 <1> ;
1346 <1> ; Volume name or long name creation
1347 <1> ; is not permitted (in TRDOS 386)!
1348 0000C869 B80B000000 <1> mov eax, ERR_FILE_ACCESS ; 11 ; 'permission denied !'
1349 0000C86E E927020000 <1> jmp sysopen_dev_err
1350 <1> ;
1351 <1> syscreat_0:
1352 <1> ;mov[u.namep], ebx
1353 0000C873 51 <1> push ecx
1354 0000C874 89DE <1> mov esi, ebx
1355 <1> ; file name is forced, change directory as temporary
1356 <1> ;mov ax, 1
1357 <1> ;mov [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
1358 <1> ;call set_working_path
1359 0000C876 E8612D0000 <1> call set_working_path_x ; 17/10/2016
1360 0000C87B 0F82D8000000 <1> jc syscreat_err
1361 <1> ;
1362 <1> ; 16/10/2016
1363 0000C881 803D[DB450100]00 <1> cmp byte [SWP_inv_fname], 0
1364 0000C888 776D <1> ja short syscreat_inv_fname ; invalid file name !

```

```

1365 <1>
1366 <1> ; Here, we have a valid path and also a valid file name
1367 <1> ; (Working dir has been changed if the path
1368 <1> ; -file name string- had contained a dir name.)
1369 <1>
1370 0000C88A 6631C0 <1> xor ax, ax
1371 <1> ;mov esi, FindFile_Name
1372 0000C88D E8BDB6FFFF <1> call find_first_file
1373 0000C892 59 <1> pop ecx
1374 <1> ; ESI = Directory Entry (FindFile_DirEntry) Location
1375 <1> ; EDI = Directory Buffer Directory Entry Location
1376 <1> ; EAX = File Size
1377 <1> ; BL = Attributes of The File/Directory
1378 <1> ; BH = Long Name Yes/No Status (>0 is YES)
1379 <1> ; DX > 0 : Ambiguous filename chars are used
1380 0000C893 726A <1> jc short syscreat_1 ; file not found (the good!)
1381 <1> ; or another error (the bad')
1382 <1>
1383 <1> ; (& the ugly!) truncate file to zero length before open
1384 <1>
1385 <1> ; '*' and '?' already checked at 'set_working_path' stage
1386 <1> ;and dx, dx
1387 <1> ;jnz short sysmkdir_err ; permission denied
1388 <1> ; invalid filename chars
1389 <1>
1390 <1> ;test cl, 10h ; subdirectory ?
1391 <1> ;jnz short sysmkdir_err
1392 <1>
1393 <1> ; BL = File Attributes:
1394 <1> ; bit 0 (1) - Read only file (R)
1395 <1> ; bit 1 (1) - Hidden file (H)
1396 <1> ; bit 2 (1) - System file (R)
1397 <1> ; bit 3 (1) - Volume label/name (V)
1398 <1> ; bit 4 (1) - Subdirectory (D)
1399 <1> ; bit 5 (1) - File has been archived
1400 <1>
1401 <1> ; * existing directory must not be truncated
1402 <1> ; (we don't know it is empty or not, at this stage)
1403 <1> ; * existing volume name (or a long name) can not be
1404 <1> ; re-created or truncated by 'syscreat'
1405 <1> ; * A file with S, H, R attributes must not be truncated
1406 <1> ; (change attributes to normal, if you need truncate it)
1407 <1>
1408 0000C895 F6C31F <1> test bl, 00011111b ; check attributes of existing file
1409 0000C898 754F <1> jnz short sysmkdir_err
1410 <1>
1411 <1> ;; normal file, OK to continue...
1412 <1>
1413 <1> ; ESI = FindFile_DirEntry
1414 0000C89A 668B4614 <1> mov ax, [esi+DirEntry_FstClusHI] ; 20
1415 0000C89E 66C1E010 <1> shl ax, 16
1416 0000C8A2 668B461A <1> mov ax, [esi+DirEntry_FstClusLO] ; 26
1417 <1> ; EAX = First cluster to be truncated/unlinked
1418 0000C8A6 57 <1> push edi
1419 0000C8A7 51 <1> push ecx
1420 0000C8A8 BE00010900 <1> mov esi, Logical_DOSDisks
1421 0000C8AD 29C9 <1> sub ecx, ecx
1422 0000C8AF 8A2D[E6380100] <1> mov ch, [Current_Drv]
1423 0000C8B5 01CE <1> add esi, ecx
1424 <1> ; ESI = Logical dos drive description table address
1425 0000C8B7 E8EBF7FFFF <1> call truncate_cluster_chain
1426 0000C8BC 59 <1> pop ecx
1427 0000C8BD 5F <1> pop edi
1428 0000C8BE 7230 <1> jc short syscreate_truncate_err
1429 <1>
1430 <1> ; 26/10/2016
1431 <1> ; EDI = Directory entry address in directory buffer
1432 <1> ; Update directory entry
1433 0000C8C0 E867DCFFFF <1> call convert_current_date_time
1434 <1> ; OUTPUT -> DX = Date in dos dir entry format
1435 <1> ; AX = Time in dos dir entry format
1436 0000C8C5 66894716 <1> mov [edi+DirEntry_WrtTime], ax
1437 0000C8C9 66895718 <1> mov [edi+DirEntry_WrtDate], dx
1438 0000C8CD 66895712 <1> mov [edi+DirEntry_LastAccDate], dx
1439 0000C8D1 31C0 <1> xor eax, eax ; file size = 0
1440 0000C8D3 89471C <1> mov [edi+DirEntry_FileSize], eax ; 0
1441 0000C8D6 C605[10410100]02 <1> mov byte [DirBuff_ValidData], 2 ; data changed sign
1442 0000C8DD BE[DC420100] <1> mov esi, FindFile_DirEntry
1443 0000C8E2 B201 <1> mov dl, 1 ; open file for writing
1444 0000C8E4 E9AA000000 <1> jmp sysopen_2
1445 <1>
1446 <1> sysmkdir_err:
1447 <1> ; 1 = write, 2 = read & write, >2 = invalid
1448 0000C8E9 B80B000000 <1> mov eax, ERR_FILE_ACCESS ; 11 ; 'permission denied !'
1449 0000C8EE EB73 <1> jmp short sysopen_err
1450 <1>
1451 <1> syscreate_truncate_err:
1452 0000C8F0 B812000000 <1> mov eax, ERR_DRV_WRITE ; 18 ; 'disk write error !'
1453 0000C8F5 EB6C <1> jmp short sysopen_err
1454 <1>
1455 <1> syscreat_inv_fname: ; invalid file name chars
1456 <1> ; 16/10/2016
1457 0000C8F7 B81A000000 <1> mov eax, ERR_INV_FILE_NAME ; 26 ; invalid file name chars
1458 0000C8FC 59 <1> pop ecx
1459 0000C8FD EB64 <1> jmp sysopen_err
1460 <1>
1461 <1> syscreat_1:
1462 <1> ; Error code in EAX
1463 0000C8FF 3C02 <1> cmp al, 02h ; 'File not found' error
1464 0000C901 7560 <1> jne sysopen_err
1465 <1>
1466 0000C903 F6C110 <1> test cl, 10h ; Directory

```

```

1467 0000C906 0F852C020000    <1>      jnz      sysmkdir_2
1468                                <1>
1469                                <1> syscreat_2:
1470 0000C90C BE[CC420100]    <1>      mov      esi, FindFile_Name
1471                                <1>      ;xor edx, edx
1472 0000C911 31C0            <1>      xor      eax, eax ; File Size = 0
1473 0000C913 31DB            <1>      xor      ebx, ebx
1474 0000C915 4B              <1>      dec      ebx ; FFFFFFFFh -> create empty file
1475                                <1>      ; (only for FAT fs)
1476                                <1>      ; CL = File Attributes
1477 0000C916 E81AECFFFF    <1>      call     create_file
1478 0000C91B 7246            <1>      jc      sysopen_err
1479                                <1>      ; EAX = New file's first cluster
1480                                <1>      ; ESI = Logical Dos Drv Descr. Table Addr.
1481                                <1>      ; EBX = offset CreateFile_Size
1482                                <1>      ; ECX = Sectors per cluster (<256)
1483                                <1>      ; EDX = Directory entry index/number (<65536)
1484                                <1>      ; 26/10/2016
1485                                <1>      ;mov  esi, Directory_Buffer
1486                                <1>      ;shl  dx, 5 ; *32
1487                                <1>      ;add  esi, edx
1488                                <1>      ;; esi = directory entry address in directory buffer
1489                                <1>
1490                                <1>      ; Here, directory entry has been created but last
1491                                <1>      ; modification date & time of the parent dir has not
1492                                <1>      ; been updated, yet!
1493                                <1>      ; (Note: Directory and FAT buffers have been updated...)
1494                                <1>
1495 0000C91D E843DDFFFF    <1>      call     update_parent_dir_lmdt ; now, it is OK too!
1496                                <1>
1497                                <1>      ; 25/10/2016
1498 0000C922 66B80018    <1>      mov      ax, 1800h
1499 0000C926 BE[CC420100]    <1>      mov      esi, FindFile_Name
1500 0000C92B E81FB6FFFF    <1>      call     find_first_file
1501 0000C930 7231            <1>      jc      short sysopen_err
1502                                <1>
1503                                <1>      ; Only possible error after here is
1504                                <1>      ; "too many open files !" error.
1505                                <1>      ;
1506                                <1>      ; If "syscreat" will return with that error,
1507                                <1>      ; (the file has been created but it could not be opened)
1508                                <1>      ; the user must retry to open this file again
1509                                <1>      ; or must close another file before using
1510                                <1>      ; "sysopen" system call.
1511                                <1>
1512 0000C932 B201            <1>      mov      dl, 1 ; open file for writing
1513                                <1>      ; ESI = Directory Entry (FindFile_DirEntry) Location
1514                                <1>      ; EAX = File Size (= 0)
1515 0000C934 EB5D            <1>      jmp      short sysopen_2
1516                                <1>
1517                                <1> sysopen: ;<open file>
1518                                <1>      ; 26/10/2016
1519                                <1>      ; 24/10/2016
1520                                <1>      ; 17/10/2016
1521                                <1>      ; 15/10/2016
1522                                <1>      ; 06/10/2016, 07/10/2016, 08/10/2016
1523                                <1>      ; 05/10/2016 (TRDOS 386 = TRDOS v2.0)
1524                                <1>      ; -derived from INT_21H.ASM-
1525                                <1>      ; ("loc_INT21h_open_file")
1526                                <1>      ; 26/02/2011
1527                                <1>      ; INT 21h Function AH = 3Dh
1528                                <1>      ; Open File
1529                                <1>      ; INPUT
1530                                <1>      ; AL= File Access Value
1531                                <1>      ; 0- Open for reading
1532                                <1>      ; 1- Open for writing
1533                                <1>      ; 2- Open for reading and writing
1534                                <1>      ; DS:DX= Pointer to filename (ASCIIIZ)
1535                                <1>      ;
1536                                <1>      ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1537                                <1>      ; 22/05/2013 - 27/05/2013 (Retro UNIX 8086 v1)
1538                                <1>      ;
1539                                <1>      ; 'sysopen' opens a file in following manner:
1540                                <1>      ; 1) The second argument in a sysopen says whether to
1541                                <1>      ; open the file ro read (0) or write (>0).
1542                                <1>      ; 2) I-node of the particular file is obtained via 'namei'.
1543                                <1>      ; 3) The file is opened by 'iopen'.
1544                                <1>      ; 4) Next housekeeping is performed on the fsp table
1545                                <1>      ; and the user's open file list - u.fp.
1546                                <1>      ; a) u.fp and fsp are scanned for the next available slot.
1547                                <1>      ; b) An entry for the file is created in the fsp table.
1548                                <1>      ; c) The number of this entry is put on u.fp list.
1549                                <1>      ; d) The file descriptor index to u.fp list is pointed
1550                                <1>      ; to by u.r0.
1551                                <1>      ;
1552                                <1>      ; Calling sequence:
1553                                <1>      ; sysopen; name; mode
1554                                <1>      ; Arguments:
1555                                <1>      ; name - file name or path name
1556                                <1>      ; mode - 0 to open for reading
1557                                <1>      ; 1 to open for writing
1558                                <1>      ; Inputs: (arguments)
1559                                <1>      ; Outputs: *u.r0 - index to u.fp list (the file descriptor)
1560                                <1>      ; is put into r0's location on the stack.
1561                                <1>      ; .....
1562                                <1>      ;
1563                                <1>      ; Retro UNIX 8086 v1 modification:
1564                                <1>      ; 'sysopen' system call has two arguments; so,
1565                                <1>      ; * 1st argument, name is pointed to by BX register
1566                                <1>      ; * 2nd argument, mode is in CX register
1567                                <1>      ;
1568                                <1>      ; AX register (will be restored via 'u.r0') will return

```

```

1569      <1>      ;      to the user with the file descriptor/number
1570      <1>      ;      (index to u.fp list).
1571      <1>      ;
1572      <1>      ;call arg2
1573      <1>      ; * name - 'u.namep' points to address of file/path name
1574      <1>      ;      in the user's program segment ('u.segmt')
1575      <1>      ;      with offset in BX register (as sysopen argument 1).
1576      <1>      ; * mode - sysopen argument 2 is in CX register
1577      <1>      ;      which is on top of stack.
1578      <1>      ;
1579      <1>      ; jsr r0,arg2 / get sys args into u.namep and on stack
1580      <1>      ;
1581      <1>      ;      system call registers: ebx, ecx (through 'sysenter')
1582      <1>      ;
1583      <1>      ; TRDOS 386 (05/10/2016)
1584      <1>      ;
1585      <1>      ; INPUT ->
1586      <1>      ;      CL = File Access Value (Open Mode)
1587      <1>      ;      0 - Open file for reading
1588      <1>      ;      1 - Open file for writing
1589      <1>      ;      2 - Open device for reading
1590      <1>      ;      3 - Open device for writing
1591      <1>      ;      EBX = Pointer to filename/devicename (ASCIIIZ)
1592      <1>      ; OUTPUT ->
1593      <1>      ;      eax = File/Device Handle/Number (index) (AL)
1594      <1>      ;      cf = 1 -> Error code in AL
1595      <1>      ;
1596      <1>      ; Modified Registers: EAX (at the return of system call)
1597      <1>      ;
1598      <1>
1599      0000C936 80F901      <1>      cmp     cl, 1 ; read file (0), write file (1)
1600      0000C939 7614      <1>      jna     short sysopen_0
1601      <1>
1602      0000C93B 80F903      <1>      cmp     cl, 3
1603      0000C93E 0F8640010000      <1>      jna     sysopen_device
1604      <1>
1605      <1>      ; Invalid access code
1606      0000C944 B817000000      <1>      mov     eax, ERR_INV_PARAMETER
1607      0000C949 0F874B010000      <1>      ja      sysopen_dev_err
1608      <1>
1609      <1> sysopen_0:
1610      <1>      ;mov     [u.namep], ebx
1611      0000C94F 51      <1>      push    ecx
1612      0000C950 89DE      <1>      mov     esi, ebx
1613      <1>      ; file name is forced, change directory as temporary
1614      <1>      ;mov     ax, 1
1615      <1>      ;mov     [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
1616      <1>      ;call    set_working_path
1617      0000C952 E8852C0000      <1>      call    set_working_path_x ; 17/10/2016
1618      0000C957 731E      <1>      jnc     short sysopen_1
1619      <1>
1620      <1> syscreat_err: ; ecx = file attributes (for 'syscreat')
1621      0000C959 59      <1>      pop     ecx ; open mode
1622      0000C95A 21C0      <1>      and     eax, eax ; 0 -> Bad Path!
1623      0000C95C 7505      <1>      jnz     short sysopen_err
1624      <1>      ; eax = 0
1625      0000C95E B80C000000      <1>      mov     eax, ERR_DIR_NOT_FOUND ; Directory not found !
1626      <1> sysopen_err:
1627      0000C963 A3[AC4E0100]      <1>      mov     [u.r0], eax
1628      0000C968 A3[104F0100]      <1>      mov     [u.error], eax
1629      0000C96D E83F2D0000      <1>      call    reset_working_path
1630      0000C972 E978FAFFFF      <1>      jmp     error
1631      <1>
1632      <1> sysopen_1:
1633      <1>      ;mov     esi, FindFile_Name
1634      0000C977 66B80018      <1>      mov     ax, 1800h ; Only files
1635      0000C97B E8CFB5FFFF      <1>      call    find_first_file
1636      0000C980 5A      <1>      pop     edx
1637      0000C981 72E0      <1>      jc      short sysopen_err ; eax = 2 (File not found !)
1638      <1>
1639      <1>      ; check_open_file_attr_access_code
1640      <1>
1641      0000C983 F6C307      <1>      test    bl, 7 ; system, hidden, readonly
1642      0000C986 740B      <1>      jz      short sysopen_2
1643      <1>
1644      0000C988 20D2      <1>      and     dl, dl ; 0 = read mode
1645      0000C98A 7407      <1>      jz      short sysopen_2
1646      <1>
1647      <1>      ; 1 = write, 2 = read & write, >2 = invalid
1648      0000C98C B80B000000      <1>      mov     eax, ERR_FILE_ACCESS ; 11 = 'permission denied !'
1649      0000C991 EBD0      <1>      jmp     short sysopen_err
1650      <1>
1651      <1> sysopen_2:
1652      <1>      ; esi = Directory Entry (FindFile_DirEntry) Location
1653      0000C993 89F3      <1>      mov     ebx, esi
1654      0000C995 31F6      <1>      xor     esi, esi ; 0
1655      0000C997 31FF      <1>      xor     edi, edi ; 0
1656      <1> sysopen_3: ; scan the list of entries in fsp table
1657      0000C999 80BE[B24E0100]00      <1>      cmp     byte [esi+u.fp], 0
1658      0000C9A0 760F      <1>      jna     short sysopen_4 ; empty slot
1659      0000C9A2 6646      <1>      inc     si
1660      0000C9A4 6683FE0A      <1>      cmp     si, 10
1661      0000C9A8 72EF      <1>      jb      short sysopen_3
1662      <1> toomanyf:
1663      0000C9AA B80D000000      <1>      mov     eax, ERR_TOO_MANY_FILES ; too many open files !
1664      0000C9AF EBB2      <1>      jmp     short sysopen_err
1665      <1>
1666      <1> sysopen_4:
1667      0000C9B1 80BF[4A490100]00      <1>      cmp     byte [edi+OF_MODE], 0 ; Scan open files table
1668      0000C9B8 760A      <1>      jna     short sysopen_5
1669      0000C9BA 6647      <1>      inc     di
1670      0000C9BC 6683FF0A      <1>      cmp     di, OPENFILES ; max. number of open files (=10)

```



```

1671 0000C9C0 72EF      <1>      jnb     short sysopen_4
1672 0000C9C2 EBE6      <1>      jmp     short toomanyf
1673                      <1>
1674                      <1> sysopen_5:
1675 0000C9C4 FEC2      <1>      inc     dl
1676 0000C9C6 8897[4A490100] <1>      mov     [edi+OF_MODE], dl
1677 0000C9CC 8A15[8A420100] <1>      mov     dl, [FindFile_Drv]
1678 0000C9D2 8897[40490100] <1>      mov     [edi+OF_DRIVE], dl ; Logical DOS drive number
1679 0000C9D8 66C1E702   <1>      shl     di, 2 ; *4 (dword offset)
1680                      <1>
1681 0000C9DC 8987[90490100] <1>      mov     [edi+OF_SIZE], eax ; File size in bytes
1682                      <1>
1683 0000C9E2 668B4314   <1>      mov     ax, [ebx+DirEntry_FstClusHI]
1684 0000C9E6 C1E010   <1>      shl     eax, 16
1685 0000C9E9 668B431A   <1>      mov     ax, [ebx+DirEntry_FstClusLO]
1686 0000C9ED 8987[18490100] <1>      mov     [edi+OF_FCLUSTER], eax ; First cluster
1687 0000C9F3 8987[304A0100] <1>      mov     [edi+OF_CCLUSTER], eax ; Current cluster
1688                      <1>
1689 0000C9F9 31DB      <1>      xor     ebx, ebx
1690 0000C9FB 899F[68490100] <1>      mov     [edi+OF_POINTER], ebx ; offset pointer (0)
1691 0000CA01 899F[584A0100] <1>      mov     [edi+OF_CCINDEX], ebx ; cluster index (0)
1692                      <1>
1693 0000CA07 A1[FC420100]   <1>      mov     eax, [FindFile_DirFirstCluster]
1694 0000CA0C 8987[B8490100] <1>      mov     [edi+OF_DIRFCLUSTER], eax
1695                      <1>
1696 0000CA12 A1[00430100]   <1>      mov     eax, [FindFile_DirCluster]
1697 0000CA17 8987[E0490100] <1>      mov     [edi+OF_DIRCLUSTER], eax
1698                      <1>
1699                      <1> ; Get (& Save) Volume ID
1700                      <1> ; Important for files of removable drives
1701                      <1> ; (In order to check the drive has same volume/disk)
1702 0000CA1D 88D7      <1>      mov     bh, dl
1703 0000CA1F 81C300010900 <1>      add     ebx, Logical_DOSDisks
1704 0000CA25 8A4303   <1>      mov     al, [ebx+LD_FATType]
1705 0000CA28 3C01      <1>      cmp     al, 1
1706 0000CA2A 7209      <1>      jnb     short sysopen_6_fs
1707 0000CA2C 3C02      <1>      cmp     al, 2
1708 0000CA2E 770A      <1>      ja      short sysopen_6_fat32
1709                      <1> sysopen_6_fat:
1710 0000CA30 8B432D   <1>      mov     eax, [ebx+LD_BPB+VolumeID]
1711 0000CA33 EB08      <1>      jmp     short sysopen_7
1712                      <1> sysopen_6_fs:
1713 0000CA35 8B4328   <1>      mov     eax, [ebx+LD_FS_VolumeSerial]
1714 0000CA38 EB03      <1>      jmp     short sysopen_7
1715                      <1> sysopen_6_fat32:
1716 0000CA3A 8B4349   <1>      mov     eax, [ebx+LD_BPB+FAT32_VolID]
1717                      <1> sysopen_7:
1718 0000CA3D A3[DC380100] <1>      mov     [Current_VolSerial], eax
1719                      <1>
1720 0000CA42 8987[084A0100] <1>      mov     [edi+OF_VOLUMEID], eax
1721                      <1>
1722                      <1> ; 24/10/2016
1723 0000CA48 66D1EF   <1>      shr     di, 1 ; 4/2, word offset
1724 0000CA4B 668B1D[04430100] <1>      mov     bx, [FindFile_DirEntryNumber]
1725 0000CA52 66899F[804A0100] <1>      mov     [edi+OF_DIRENTRY], bx
1726                      <1>
1727 0000CA59 31D2      <1>      xor     edx, edx
1728                      <1> ;shr di, 2 ; /4 (byte offset)
1729 0000CA5B 66D1EF   <1>      shr     di, 1 ; 2/2, byte offset
1730 0000CA5E 8897[5E490100] <1>      mov     byte [edi+OF_OPENCOUNT], dl ; 0
1731 0000CA64 8897[54490100] <1>      mov     byte [edi+OF_STATUS], dl ; 0
1732                      <1>
1733 0000CA6A 89FB      <1>      mov     ebx, edi
1734 0000CA6C FEC3      <1>      inc     bl
1735                      <1>
1736 0000CA6E 889E[B24E0100] <1>      mov     [esi+u.fp], bl ; Open File Entry Number
1737 0000CA74 8935[AC4E0100] <1>      mov     [u.r0], esi ; move index to u.fp list
1738                      <1> ; into eax on stack
1739                      <1>
1740 0000CA7A E8322C0000 <1>      call    reset_working_path
1741                      <1>
1742 0000CA7F E98BF9FFFF <1>      jmp     sysret
1743                      <1>
1744                      <1> ; (Retro UNIX 386 v1.0)
1745                      <1> ; 'fsp' table (10 bytes/entry)
1746                      <1> ; bit 15 bit 0
1747                      <1> ; ---|-----
1748                      <1> ; r/w| i-number of open file
1749                      <1> ; ---|-----
1750                      <1> ; device number
1751                      <1> ; -----
1752                      <1> ; offset pointer, r/w pointer to file (bit 0-15)
1753                      <1> ; -----
1754                      <1> ; offset pointer, r/w pointer to file (bit 16-31)
1755                      <1> ; -----|-----
1756                      <1> ; flag that says file | number of processes
1757                      <1> ; has been deleted | that have file open
1758                      <1> ; -----|-----
1759                      <1>
1760                      <1> sysopen_device:
1761                      <1> ; 15/10/2016
1762                      <1> ; 08/10/2016
1763                      <1> ; 07/10/2016 (TRDOS 386 = TRDOS v2.0)
1764 0000CA84 51      <1>      push    ecx ; open mode
1765 0000CA85 89E5      <1>      mov     ebp, esp
1766 0000CA87 B910000000 <1>      mov     ecx, 16 ; transfer length = 16 bytes
1767 0000CA8C 29CC      <1>      sub     esp, ecx
1768 0000CA8E 89E7      <1>      mov     edi, esp ; destination address
1769 0000CA90 89DE      <1>      mov     esi, ebx ; dev name in user's memory space
1770 0000CA92 E8831D0000 <1>      call    transfer_from_user_buffer
1771 0000CA97 7310      <1>      jnc     short sysopen_dev_0
1772                      <1> ; eax = ERR_OUT_OF_MEMORY = 4 = ERR_MINOR_IM

```

```

1773 0000CA99 59          <1>      pop     ecx
1774                      <1> sysopen_dev_err:
1775 0000CA9A A3[AC4E0100] <1>      mov     [u.r0], eax
1776 0000CA9F A3[104F0100] <1>      mov     [u.error], eax
1777 0000CAA4 E946F9FFFF <1>      jmp     error
1778                      <1> sysopen_dev_0:
1779 0000CAA9 89FE <1>      mov     esi, edi ; Device name addr (max. 16 bytes, ASCIIIZ)
1780                      <1>      ; for example: "tty, TTY, /dev/tty"
1781 0000CAAB E8A92E0000 <1>      call    get_device_number
1782 0000CAB0 89EC <1>      mov     esp, ebp
1783 0000CAB2 59 <1>      pop     ecx
1784 0000CAB3 7307 <1>      jnc     short sysopen_dev_1
1785 0000CAB5 B818000000 <1>      mov     eax, ERR_INV_DEV_NAME ; 24 ; 'invalid device name !'
1786 0000CABA EBDE <1>      jmp     short sysopen_dev_err
1787                      <1> sysopen_dev_1:
1788                      <1>      ; eax = Device Number (AL)
1789                      <1>      ; cl = Open mode (2 = device read, 3 = device write)
1790 0000CABC 31DB <1>      xor     ebx, ebx ; 0
1791                      <1> sysopen_dev_2: ; scan the list of entries
1792 0000CABE 389B[B24E0100] <1>      cmp     [ebx+u.fpl], bl ; 0
1793 0000CAC4 760E <1>      jna     short sysopen_dev_3 ; empty slot
1794 0000CAC6 FEC3 <1>      inc     bl
1795 0000CAC8 80FB0A <1>      cmp     bl, 10
1796 0000CACB 72F1 <1>      jnb     short sysopen_dev_2
1797                      <1>      ;
1798 0000CACD B80D000000 <1>      mov     eax, ERR_TOO_MANY_FILES ; too many open files !
1799 0000CAD2 EBC6 <1>      jmp     short sysopen_dev_err
1800                      <1> sysopen_dev_3:
1801 0000CAD4 891D[AC4E0100] <1>      mov     [u.r0], ebx ; File/Device index/handle/descriptor
1802                      <1>      ; eax = device number (entry offset)
1803 0000CADA 8AA8[DC460100] <1>      mov     ch, [eax+DEV_ACCESS] ; bit 0 = accessable by users
1804                      <1>      ; bit 1 = read access perm
1805                      <1>      ; bit 2 = write access perm
1806                      <1>      ; bit 3 = IOCTL permit to users
1807                      <1>      ; bit 4 = block device if set
1808                      <1>      ; bit 5 = 16 bit or 1024 byte
1809                      <1>      ; bit 6 = 32 bit or 2048 byte
1810                      <1>      ; bit 7 = installable device drv
1811 0000CAE0 F6C501 <1>      test    ch, 1 ; accessable by normal users (except root)
1812 0000CAE3 7510 <1>      jnz     short sysopen_dev_4 ; yes, permission has been given
1813 0000CAE5 803D[F84E0100]00 <1>      cmp     byte [u.uid], 0 ; root?
1814 0000CAEC 7607 <1>      jna     short sysopen_dev_4 ; superuser can open all devices
1815                      <1> sysopen_dev_perm_err:
1816 0000CAEE B80B000000 <1>      mov     eax, ERR_DEV_ACCESS ; 11 = 'permission denied !'
1817 0000CAF3 EBA5 <1>      jmp     short sysopen_dev_err
1818                      <1> sysopen_dev_4:
1819 0000CAF5 D0ED <1>      shr     ch, 1 ; result: 1 = read, 2 = write, 3 = r & w
1820 0000CAF7 FEC9 <1>      dec     cl ; result: 1 = read, 2 = write
1821 0000CAF9 84E9 <1>      test    cl, ch
1822 0000CAFB 74F1 <1>      jz     short sysopen_dev_perm_err
1823                      <1>      ;
1824 0000CAFD D0E5 <1>      shl     ch, 1 ; bit 0 = 0
1825                      <1>      ; eax = device number (entry offset)
1826 0000CAFF E8712F0000 <1>      call    device_open
1827 0000CB04 72E8 <1>      jc     short sysopen_dev_perm_err
1828                      <1>      ;
1829                      <1>      ; eax = device number (entry offset)
1830 0000CB06 0C80 <1>      or      al, 80h ; set device bit (set bit 7 to 1)
1831 0000CB08 8B1D[AC4E0100] <1>      mov     ebx, [u.r0]
1832 0000CB0E 8883[B24E0100] <1>      mov     [ebx+u.fpl], al ; bit 7 (=1) points to device
1833                      <1>      ;
1834 0000CB14 E9F6F8FFFF <1>      jmp     sysret
1835                      <1>      ;
1836                      <1> sysmkdir: ; < make directory >
1837                      <1>      ; 15/10/2016
1838                      <1>      ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)
1839                      <1>      ; -derived from INT_21H.ASM-
1840                      <1>      ; ("loc_INT21h_create_file")
1841                      <1>      ; 10/07/2011 (12/03/2011)
1842                      <1>      ; INT 21h Function AH = 3Ch
1843                      <1>      ; Create File
1844                      <1>      ; INPUT
1845                      <1>      ; CX = Attributes
1846                      <1>      ; DS:DX= Address of zero terminaned path name
1847                      <1>      ;
1848                      <1>      ;
1849                      <1>      ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1850                      <1>      ; 27/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
1851                      <1>      ;
1852                      <1>      ; 'sysmkdir' creates an empty directory whose name is
1853                      <1>      ; pointed to by arg 1. The mode of the directory is arg 2.
1854                      <1>      ; The special entries '.' and '..' are not present.
1855                      <1>      ; Errors are indicated if the directory already exists or
1856                      <1>      ; user is not the super user.
1857                      <1>      ;
1858                      <1>      ; Calling sequence:
1859                      <1>      ; sysmkdir; name; mode
1860                      <1>      ; Arguments:
1861                      <1>      ; name - points to the name of the directory
1862                      <1>      ; mode - mode of the directory
1863                      <1>      ; Inputs: (arguments)
1864                      <1>      ; Outputs: -
1865                      <1>      ; (sets 'directory' flag to 1;
1866                      <1>      ; 'set user id on execution' and 'executable' flags to 0)
1867                      <1>      ; .....
1868                      <1>      ;
1869                      <1>      ; Retro UNIX 8086 v1 modification:
1870                      <1>      ; 'sysmkdir' system call has two arguments; so,
1871                      <1>      ; * 1st argument, name is pointed to by BX register
1872                      <1>      ; * 2nd argument, mode is in CX register
1873                      <1>      ;
1874                      <1>      ; TRDOS 386 (10/10/2016)

```

```

1875 <1> ;
1876 <1> ; INPUT ->
1877 <1> ; CL = Directory Attributes
1878 <1> ; bit 0 (1) - Read only file/dir (R)
1879 <1> ; bit 1 (1) - Hidden file/dir (H)
1880 <1> ; bit 2 (1) - System file/dir (R)
1881 <1> ; bit 3 (1) - Volume label/name (V)
1882 <1> ; bit 4 (1) - Subdirectory (D)
1883 <1> ; bit 5 (1) - File/Dir has been archived (A)
1884 <1> ; CX = 0 -> create normal directory
1885 <1> ; EBX = Pointer to directory name (ASCIIIZ) -path-
1886 <1> ;
1887 <1> ; OUTPUT ->
1888 <1> ; eax = First cluster of the new directory
1889 <1> ; cf = 1 -> Error code in AL
1890 <1> ;
1891 <1> ; Modified Registers: EAX (at the return of system call)
1892 <1> ;
1893 <1> ; Note: If the file or directory is existing
1894 <1> ; an access error will be returned.
1895 <1>
1896 0000CB19 6621C9 <1> and cx, cx ; if cx = 0 -> create a normal subdir
1897 0000CB1C 7413 <1> jz short sysmkdir_1
1898 <1>
1899 0000CB1E F6C110 <1> test cl, 10h ; if dir flags set, also use other flags
1900 0000CB21 0F853DFDFFFF <1> jnz sysmkdir_0 ; jump to head of 'syscreat'
1901 <1>
1902 <1> ; CX has wrong flags
1903 0000CB27 B817000000 <1> mov eax, ERR_INV_FLAGS
1904 0000CB2C E969FFFFFF <1> jmp sysopen_dev_err
1905 <1>
1906 <1> sysmkdir_1:
1907 0000CB31 B110 <1> mov cl, 10h ; set subdir flag and reset other flags
1908 0000CB33 E92CFDFFFF <1> jmp sysmkdir_0 ; jump to head of 'syscreat'
1909 <1> sysmkdir_2:
1910 <1> ; jump from 'syscreat' ; from 'syscreat_1'
1911 <1> ; CL = Directory attributes/flags
1912 0000CB38 BE[CC420100] <1> mov esi, FindFile_Name
1913 0000CB3D E823D7FFFF <1> call make_sub_directory
1914 0000CB42 0F821BFEFFFF <1> jc sysopen_err ; NOTE: Old type (TRDOS 8086)
1915 <1> ; error codes must be modified
1916 <1> ; for next TRDOS 386 versions
1917 <1> ; (10/10/2016)
1918 <1> ; Old (MSDOS type)
1919 <1> ; error codes (2011):
1920 <1> ; 2 = file not found
1921 <1> ; 3 = directory not found
1922 <1> ; 5 = access denied
1923 <1> ; 12 = no more files
1924 <1> ; 19 = disk write protected
1925 <1> ; 39 = insufficient disk space
1926 <1> ; 'sysdefs.s' ; 10/10/2016
1927 <1>
1928 0000CB48 A3[AC4E0100] <1> mov [u.r0], eax ; New sub dir's first cluster
1929 <1>
1930 0000CB4D E85F2B0000 <1> call reset_working_path
1931 <1>
1932 0000CB52 E9B8F8FFFF <1> jmp sysret
1933 <1>
1934 <1> sysclose: ;<close file>
1935 <1> ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
1936 <1> ;
1937 <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1938 <1> ; 22/05/2013 - 26/05/2013 (Retro UNIX 8086 v1)
1939 <1> ;
1940 <1> ; 'sysclose', given a file descriptor in 'u.r0', closes the
1941 <1> ; associated file. The file descriptor (index to 'u.fp' list)
1942 <1> ; is put in r1 and 'fclose' is called.
1943 <1> ;
1944 <1> ; Calling sequence:
1945 <1> ; sysclose
1946 <1> ; Arguments:
1947 <1> ; -
1948 <1> ; Inputs: *u.r0 - file descriptor
1949 <1> ; Outputs: -
1950 <1> ; .....
1951 <1> ;
1952 <1> ; Retro UNIX 8086 v1 modification:
1953 <1> ; The user/application program puts file descriptor
1954 <1> ; in BX register as 'sysclose' system call argument.
1955 <1> ; (argument transfer method 1)
1956 <1>
1957 <1> ; TRDOS 386 (06/10/2016)
1958 <1> ;
1959 <1> ; INPUT ->
1960 <1> ; EBX = File Handle/Number (file index) (AL)
1961 <1> ; OUTPUT ->
1962 <1> ; cf = 0 -> EAX = 0
1963 <1> ; cf = 1 -> Error code in EAX (ERR_FILE_NOT_OPEN)
1964 <1> ;
1965 <1> ; Modified Registers: EAX (at the return of system call)
1966 <1> ;
1967 <1>
1968 0000CB57 89D8 <1> mov eax, ebx
1969 0000CB59 31DB <1> xor ebx, ebx
1970 0000CB5B 891D[AC4E0100] <1> mov [u.r0], ebx ; 0 ; return value of EAX
1971 0000CB61 E89D0E0000 <1> call fclose
1972 0000CB66 0F83A3F8FFFF <1> jnc sysret
1973 0000CB6C B80A000000 <1> mov eax, ERR_FILE_NOT_OPEN ; file not open !
1974 0000CB71 A3[104F0100] <1> mov [u.error], eax ;
1975 0000CB76 A3[AC4E0100] <1> mov [u.r0], eax ; ! invalid handle !
1976 0000CB7B E96FF8FFFF <1> jmp error

```

```

1977 <1>
1978 <1> sysread: ; < read from file >
1979 <1> ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
1980 <1> ; -derived from INT_21H.ASM-
1981 <1> ; ("loc_INT21h_read_file")
1982 <1> ; 13/03/2011 (05/03/2011)
1983 <1> ; INT 21h Function AH = 3Fh
1984 <1> ; Read from a File
1985 <1> ; INPUT
1986 <1> ; BX = File Handle
1987 <1> ; CX = Number of bytes to read
1988 <1> ; DS:DX= Buffer address
1989 <1> ;
1990 <1> ; Note: TRDOS 386 'sysread' has been derived from
1991 <1> ; Retro UNIX 386 v1 'sysread', except a few
1992 <1> ; code modifications.
1993 <1> ;
1994 <1> ; 13/05/2015 (Retro UNIX 386 v1)
1995 <1> ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
1996 <1> ; 23/05/2013 (Retro UNIX 8086 v1)
1997 <1> ;
1998 <1> ; 'sysread' is given a buffer to read into and the number of
1999 <1> ; characters to be read. If finds the file from the file
2000 <1> ; descriptor located in *u.r0 (r0). This file descriptor
2001 <1> ; is returned from a successful open call (sysopen).
2002 <1> ; The i-number of file is obtained via 'rw1' and the data
2003 <1> ; is read into core via 'readi'.
2004 <1> ;
2005 <1> ; Calling sequence:
2006 <1> ; sysread; buffer; nchars
2007 <1> ; Arguments:
2008 <1> ; buffer - location of contiguous bytes where
2009 <1> ; input will be placed.
2010 <1> ; nchars - number of bytes or characters to be read.
2011 <1> ; Inputs: *u.r0 - file descriptor (& arguments)
2012 <1> ; Outputs: *u.r0 - number of bytes read.
2013 <1> ; .....
2014 <1> ;
2015 <1> ; Retro UNIX 8086 v1 modification:
2016 <1> ; 'sysread' system call has three arguments; so,
2017 <1> ; * 1st argument, file descriptor is in BX register
2018 <1> ; * 2nd argument, buffer address/offset in CX register
2019 <1> ; * 3rd argument, number of bytes is in DX register
2020 <1> ;
2021 <1> ; AX register (will be restored via 'u.r0') will return
2022 <1> ; to the user with number of bytes read.
2023 <1> ;
2024 <1> ; TRDOS 386 (05/10/2016)
2025 <1> ;
2026 <1> ; INPUT ->
2027 <1> ; EBX = File handle (descriptor/index)
2028 <1> ; ECX = Buffer address
2029 <1> ; EDX = Number of bytes
2030 <1> ; OUTPUT ->
2031 <1> ; EAX = Number of bytes have been read
2032 <1> ; cf = 1 -> Error code in AL
2033 <1> ;
2034 <1> ; Modified Registers: EAX (at the return of system call)
2035 <1> ;
2036 <1>
2037 <1> ; EBX = File descriptor
2038 0000CB80 E8CC0E0000 <1> call getfl
2039 0000CB85 7277 <1> jc short device_read ; read data from device
2040 <1> ; EAX = First cluster of the file
2041 <1>
2042 0000CB87 E83F000000 <1> call rw1
2043 0000CB8C 730A <1> jnc short sysread_0
2044 <1>
2045 0000CB8E A3[AC4E0100] <1> mov [u.r0], eax ; error code
2046 0000CB93 E957F8FFFF <1> jmp error
2047 <1>
2048 <1> sysread_0:
2049 0000CB98 E85A170000 <1> call readi
2050 0000CB9D EB1D <1> jmp short rw0
2051 <1>
2052 <1> syswrite: ; < write to file >
2053 <1> ; 23/10/2016
2054 <1> ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2055 <1> ; -derived from INT_21H.ASM-
2056 <1> ; ("loc_INT21h_write_file")
2057 <1> ; 13/03/2011 (05/03/2011)
2058 <1> ; INT 21h Function AH = 40h
2059 <1> ; Write to a File
2060 <1> ; INPUT
2061 <1> ; BX = File Handle
2062 <1> ; CX = Number of bytes to write
2063 <1> ; DS:DX= Buffer address
2064 <1> ;
2065 <1> ; Note: TRDOS 386 'sysrwrite' has been derived from
2066 <1> ; Retro UNIX 386 v1 'syswrite', except a few
2067 <1> ; code modifications.
2068 <1> ;
2069 <1>
2070 <1> ; 13/05/2015 (Retro UNIX 386 v1)
2071 <1> ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
2072 <1> ; 23/05/2013 (Retro UNIX 8086 v1)
2073 <1> ;
2074 <1> ; 'syswrite' is given a buffer to write onto an output file
2075 <1> ; and the number of characters to write. If finds the file
2076 <1> ; from the file descriptor located in *u.r0 (r0). This file
2077 <1> ; descriptor is returned from a successful open or create call
2078 <1> ; (sysopen or syscreat). The i-number of file is obtained via

```



```

2079      <1>      ; 'rw1' and buffer is written on the output file via 'write'.
2080      <1>      ;
2081      <1>      ; Calling sequence:
2082      <1>      ;      syswrite; buffer; nchars
2083      <1>      ; Arguments:
2084      <1>      ;      buffer - location of contiguous bytes to be writtten.
2085      <1>      ;      nchars - number of characters to be written.
2086      <1>      ; Inputs: *u.r0 - file descriptor (& arguments)
2087      <1>      ; Outputs: *u.r0 - number of bytes written.
2088      <1>      ; .....
2089      <1>      ;
2090      <1>      ; Retro UNIX 8086 v1 modification:
2091      <1>      ;      'syswrite' system call has three arguments; so,
2092      <1>      ;      * 1st argument, file descriptor is in BX register
2093      <1>      ;      * 2nd argument, buffer address/offset in CX register
2094      <1>      ;      * 3rd argument, number of bytes is in DX register
2095      <1>      ;
2096      <1>      ;      AX register (will be restored via 'u.r0') will return
2097      <1>      ;      to the user with number of bytes written.
2098      <1>      ;
2099      <1>      ; INPUT ->
2100      <1>      ;      EBX = File handle (descriptor/index)
2101      <1>      ;      ECX = Buffer address
2102      <1>      ;      EDX = Number of bytes
2103      <1>      ; OUTPUT ->
2104      <1>      ;      EAX = Number of bytes have been written
2105      <1>      ;      cf = 1 -> Error code in AL
2106      <1>      ;
2107      <1>      ; Modified Registers: EAX (at the return of system call)
2108      <1>      ;
2109      <1>      ;
2110      <1>      ; EBX = File descriptor
2111      <1>      call    getf1
2112      <1>      jc     short device_write ; write data to device
2113      <1>      ; EAX = First cluster of the file
2114      <1>      ; EBX = File number (Open file number) ; 23/10/2016
2115      <1>      ;
2116      <1>      call    rw1
2117      <1>      jnc    short syswrite_0
2118      <1>      mov     [u.r0], eax ; error code
2119      <1>      jmp     error
2120      <1>      ;
2121      <1>      syswrite_0:
2122      <1>      call    writei
2123      <1>      rw0: ; 1:
2124      <1>      mov     eax, [u.nread]
2125      <1>      mov     [u.r0], eax
2126      <1>      jmp     sysret
2127      <1>      ;
2128      <1>      rw1:
2129      <1>      ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2130      <1>      ; 14/05/2015 (Retro UNIX 386 v1)
2131      <1>      ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
2132      <1>      ; 23/05/2013 - 24/05/2013 (Retro UNIX 8086 v1)
2133      <1>      ; System call registers: ebx, ecx, edx (through 'sysenter')
2134      <1>      ;
2135      <1>      ; EBX = File descriptor
2136      <1>      ; call getf1 ; calling point in 'getf' from 'rw1'
2137      <1>      ; jc short device_rw ; read/write data from/to device
2138      <1>      ; EAX = First cluster of the file
2139      <1>      ;
2140      <1>      cmp     eax, 2
2141      <1>      jnb     short rw2
2142      <1>      ;
2143      <1>      mov     [u.base], ecx ; buffer address/offset
2144      <1>      ; (in the user's virtual memory space)
2145      <1>      mov     [u.count], edx
2146      <1>      ;
2147      <1>      mov     dword [u.error], 0 ; reset the last error code
2147      <1>      <1>      0000CBE4 0000
2148      <1>      retn
2149      <1>      ;
2150      <1>      rw2:
2151      <1>      mov     eax, ERR_FILE_NOT_OPEN ; file not open !
2152      <1>      mov     dword [u.error], eax
2153      <1>      retn
2154      <1>      rw3:
2155      <1>      mov     eax, ERR_FILE_ACCESS ; permission denied !
2156      <1>      mov     dword [u.error], eax
2157      <1>      stc
2158      <1>      retn
2159      <1>      ;
2160      <1>      device_read:
2161      <1>      ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2162      <1>      ; cl = DEV_OPENMODE ; open mode
2163      <1>      ; ch = DEV_ACCESS ; access flags
2164      <1>      ; al = DEV_DRIVER ; device number (eax)
2165      <1>      ;
2166      <1>      test    cl, 1 ; 1 = read, 2 = write, 3 = read&write
2167      <1>      jz     short rw3
2168      <1>      ;
2169      <1>      mov     ebx, eax
2170      <1>      shl     bx, 2 ; *4
2171      <1>      ;
2172      <1>      test    ch, 80h ; bit 7, installable device driver flag
2173      <1>      jz     short d_read_2 ; Kernel device
2174      <1>      ; installable device
2175      <1>      d_read_1:
2176      <1>      jmp     dword [ebx+IDEV_RADDR-4]
2177      <1>      d_read_2:
2178      <1>      jmp     dword [ebx+KDEV_RADDR-4]
2179      <1>      ;

```

```

2180 <1> device_write:
2181 <1> ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2182 <1> ; cl = DEV_OPENMODE ; open mode
2183 <1> ; ch = DEV_ACCESS ; access flags
2184 <1> ; al = DEV_DRIVER ; device number (eax)
2185 <1>
2186 0000CC1A F6C102 <1> test cl, 2 ; 1 = read, 2 = write, 3 = read&write
2187 0000CC1D 74D3 <1> jz short rw3
2188 <1>
2189 0000CC1F 89C3 <1> mov ebx, eax
2190 0000CC21 66C1E302 <1> shl bx, 2 ; *4
2191 <1>
2192 0000CC25 F6C580 <1> test ch, 80h ; bit 7, installable device driver flag
2193 0000CC28 7406 <1> jz short d_write_2 ; Kernel device
2194 <1> ; installable device
2195 <1> d_write_1:
2196 0000CC2A FFA3[B8460100] <1> jmp dword [ebx+IDEV_WADDR-4]
2197 <1> d_write_2:
2198 0000CC30 FFA3[74030100] <1> jmp dword [ebx+KDEV_WADDR-4]
2199 <1>
2200 <1>
2201 <1> sysemt: ; enable (or disable) multi tasking -time sharing-
2202 <1> ;
2203 <1> ; 23/05/2016 - TRDOS 386 (TRDOS v2.0)
2204 <1> ; 14/05/2015 (Retro UNIX 386 v1)
2205 <1> ; 10/12/2013 - 20/04/2014 (Retro UNIX 8086 v1)
2206 <1> ;
2207 <1> ; Retro UNIX 8086 v1 modification:
2208 <1> ; 'Enable Multi Tasking' system call instead
2209 <1> ; of 'Emulator Trap' in original UNIX v1 for PDP-11.
2210 <1> ;
2211 <1> ; Retro UNIX 8086 v1 feature only!
2212 <1> ; Using purpose: Kernel will start without time-out
2213 <1> ; (internal clock/timer) functionality.
2214 <1> ; Then etc/init will enable clock/timer for
2215 <1> ; multi tasking.
2216 <1> ;
2217 <1> ; INPUT ->
2218 <1> ; BL = 0 -> disable multi tasking
2219 <1> ; BL > 1 -> enable multi tasking (time sharing)
2220 <1> ; OUTPUT ->
2221 <1> ; none
2222 <1> ;
2223 <1> ; Note: Multi tasking is disabled during system
2224 <1> ; initialization, it must be enabled by using
2225 <1> ; this system call. (Otherwise, running proces
2226 <1> ; will not be changed by another process within
2227 <1> ; run time sequence/schedule, if running process
2228 <1> ; will not 'release' itself. Only 'wakeup' procedure
2229 <1> ; for waiting processes and programmed timer events
2230 <1> ; for other processes can change running process
2231 <1> ; while multi tasking is disabled.) ** 23/05/2016 **
2232 <1>
2233 0000CC36 803D[F84E0100]00 <1> cmp byte [u.uid], 0 ; root ?
2234 <1> ;ja error
2235 0000CC3D 0F87F1F8FFFF <1> ja badsys ; 14/05/2015
2236 <1> ;
2237 0000CC43 FA <1> cli
2238 0000CC44 881D[B6450100] <1> mov [multi_tasking], bl ; 0 to disable, >0 to enable
2239 0000CC4A E9C0F7FFFF <1> jmp sysret
2240 <1>
2241 <1> systimer:
2242 <1> ; 02/01/2017
2243 <1> ; 21/12/2016
2244 <1> ; 19/12/2016
2245 <1> ; 10/12/2016 (callback)
2246 <1> ; 10/06/2016
2247 <1> ; 07/06/2016
2248 <1> ; 06/06/2016
2249 <1> ; 21/05/2016
2250 <1> ; 19/05/2016
2251 <1> ; 18/05/2016 - TRDOS 386 (TRDOS v2.0)
2252 <1> ; (TRDOS 386 feature only!)
2253 <1> ;
2254 <1> ; (start or stop timer event(s))
2255 <1> ;
2256 <1> ; INPUT ->
2257 <1> ; BL = Signal return byte (response byte)
2258 <1> ; (Any requested value between 0 and 255)
2259 <1> ; (Kernel will put it at the requested address)
2260 <1> ; BH = Time count unit
2261 <1> ; 0 = Stop timer event
2262 <1> ; 1 = 18.2 ticks per second
2263 <1> ; 2 = 10 milliseconds
2264 <1> ; 3 = 1 second (for real time clock interrupt)
2265 <1> ; 4 = time/tick count in current time count unit
2266 <1> ; // 10/12/2016
2267 <1> ; 80h = Stop timer event (callback method)
2268 <1> ; 81h = 18.2 ticks per second, callback method
2269 <1> ; 82h = 10 milliseconds, callback method
2270 <1> ; 83h = 1 second (for RTC int), callback method
2271 <1> ; 84h = current time count unit, callback method
2272 <1> ;
2273 <1> ; Note: Only 03h or 83h will set real time clock
2274 <1> ; (RTC) events (Others are for PIT events)!
2275 <1> ;
2276 <1> ; NOTE: If callback (user service) method is used,
2277 <1> ; EDX will point to the return address (of service
2278 <1> ; procedure) in user's space instead of signal
2279 <1> ; response byte address. (TRDOS 386 kernel will
2280 <1> ; direct the cpu to that address -in user's space-
2281 <1> ; at the return of system call or interrupt

```

```

2282      <1>      ;      just after the adjusted count/time is elapsed.)
2283      <1>      ;
2284      <1>      ;      User's sevice routine must be ended with a
2285      <1>      ;      'iret'. Normal return addresses from system
2286      <1>      ;      calls or and interrupts will be kept same except
2287      <1>      ;      the timer returns.
2288      <1>      ;
2289      <1>      ;      BH = 0 -> Stop timer event
2290      <1>      ;      BL = Timer event number (1 to 255) if BH = 0
2291      <1>      ;      If BL = 0, all timer events (which are belongs
2292      <1>      ;      to running process) will be stopped
2293      <1>      ;      ECX = Time/Tick count (depending on time count unit)
2294      <1>      ;      EDX = Signal return (Response) byte address
2295      <1>      ;      (virtual address in user's memory space)
2296      <1>      ;      OUTPUT ->
2297      <1>      ;      AL = Timer event number (1 to 255) (max. value = 16)
2298      <1>      ;      IF BH Input = 0 & CF = 0 & AL = 0 ->
2299      <1>      ;      timer event(s) has/have been stopped/finished
2300      <1>      ;      CF = 1 & AL = 0 -> no timer setting space to set
2301      <1>      ;      CF = 1 & AL > 0 -> timer count unit is not usable
2302      <1>      ;
2303      <1>      ;      NOTE: To modify a time count for a user function,
2304      <1>      ;      at first, current timer event must be stopped
2305      <1>      ;      then a new timer event (which is related with
2306      <1>      ;      same user function) must be started.
2307      <1>      ;
2308      <1>      ;      Signal return (response) byte may be used for
2309      <1>      ;      several purposes. Kernel will put this value
2310      <1>      ;      to requested address during timer interrupt,
2311      <1>      ;      program/user can check this value to understand
2312      <1>      ;      which event has been occurred and what is changed.
2313      <1>      ;      (Multi timer events can share same signal address)
2314      <1>      ;
2315      <1>      ;      NOTE: If the process is running while the time count
2316      <1>      ;      is reached, kernel will put signal return (response)
2317      <1>      ;      byte value at requested address during timer
2318      <1>      ;      interrupt and the process will continue to run.
2319      <1>      ;      Program/process must call (jump to) it's timer event
2320      <1>      ;      function as required, for checking the timer event
2321      <1>      ;      status via signal return (response) byte address.
2322      <1>      ;
2323      <1>      ;      If the process is not running (waiting or sleeping
2324      <1>      ;      or released) while the time count is reached,
2325      <1>      ;      it is restarted from where it left, to ensure
2326      <1>      ;      proper multi media (video, audio, clock, timer)
2327      <1>      ;      functionality.
2328      <1>      ;
2329      <1>      ;      (It is better to use 'syswait' or 'sysssleep',
2330      <1>      ;      or 'sysrele' system call just after the timer
2331      <1>      ;      function. Otherwise, timer events may block other
2332      <1>      ;      processes which are not using timer events.)
2333
2332      <1>      ;
2333      <1>      ;      Timer Event Structure: (max. 16 timer events, 16*16 bytes)
2334      <1>      ;      Owner:          resb 1 ; 0 = free
2335      <1>      ;                      ;>0 = process number (u.uno)
2336      <1>      ;      Calback:      resb 1 ; 1 = callback, 0 = response byte
2337      <1>      ;      Interrupt:    resb 1 ; 0 = Timer interrupt (or none)
2338      <1>      ;                      ; 1 = Real Time Clock interrupt
2339      <1>      ;      Response:     resb 1 ; 0 to 255, signal return value
2340      <1>      ;      Count Limit: resd 1 ; count of ticks (total/set)
2341      <1>      ;      Current Count: resd 1 ; count of ticks (current)
2342      <1>      ;      Response Addr: resd 1 ; response byte (pointer) address
2343      <1>      ;
2344      <1>      ;
2345      <1>      ;      19/12/2016 (timer callback)
2346      0000CC4F C605[F44A0100]00      <1>      mov     byte [tcallback], 0
2347      0000CC56 C605[F54A0100]00      <1>      mov     byte [trtc], 0
2348      0000CC5D C705[184F0100]0000-   <1>      mov     dword [u.tcb], 0 ; this is not necessary...
2348      0000CC65 0000                  <1>
2349      <1>
2350      0000CC67 80FF80                  <1>      cmp     bh, 80h
2351      0000CC6A 7225                  <1>      jnb     short systimer_cb2
2352      0000CC6C 7704                  <1>      ja      short systimer_cb0
2353      <1>
2354      0000CC6E 31D2                  <1>      xor     edx, edx ; 0, reset callback address
2355      0000CC70 EB0B                  <1>      jmp     short systimer_cb1
2356      <1>
2357      <1>      systimer_cb0:
2358      0000CC72 80FF84                  <1>      cmp     bh, 84h
2359      0000CC75 7764                  <1>      ja      short systimer_5 ; undefined, error
2360      <1>
2361      <1>      ;mov     byte [tcallback], 1 ; 19/12/2016
2362      0000CC77 FE05[F44A0100]          <1>      inc     byte [tcallback]
2363      <1>
2364      <1>      systimer_cb1:
2365      0000CC7D 0FB635[FB4E0100]          <1>      movzx   esi, byte [u.uno] ; process number
2366      0000CC84 66C1E602                <1>      shl     si, 2
2367      0000CC88 8996[524C0100]          <1>      mov     [esi+p.tcb-4], edx ; set process timer callback address
2368      <1>      ; (overwrite prev value if it is set!)
2369      0000CC8E 80E77F                  <1>      and     bh, 7Fh
2370      <1>
2371      <1>      systimer_cb2:
2372      0000CC91 80FF02                  <1>      cmp     bh, 2
2373      0000CC94 7445                  <1>      je      short systimer_5 ; only 18.2 ticks per second is usable
2374      <1>      ; 10 milliseconds (100 Hertz) timer
2375      <1>      ; will be set later (18/05/2016)
2376      0000CC96 774B                  <1>      ja      short systimer_6
2377      <1>
2378      0000CC98 20FF                  <1>      and     bh, bh
2379      0000CC9A 0F84BA000000            <1>      jz      systimer_9 ; stop timer event(s)
2380      <1>
2381      <1>      ; bh = 1 (timer interrupt, 18.2 Hz, IBM PC/AT ROMBIOS default)

```

```

2382 <1>
2383 <1> systimer_19:
2384 0000CCA0 B00A <1> mov al, 10 ; (*)
2385 <1>
2386 <1> systimer_0:
2387 0000CCA2 B710 <1> mov bh, 16
2388 <1> ;
2389 0000CCA4 383D[B7450100] <1> cmp [timer_events], bh ; 16 ; 07/06/2016
2390 0000CCAA 7319 <1> jnb short systimer_3 ; max. 16 timer events
2391 <1> ;
2392 0000CCAC 50 <1> push eax ; (*)
2393 <1>
2394 0000CCAD BF[A84F0100] <1> mov edi, timer_set ; beginning address of timer events
2395 <1> ; setting space
2396 0000CCB2 30C0 <1> xor al, al ; 0
2397 <1> systimer_1:
2398 0000CCB4 FEC0 <1> inc al
2399 0000CCB6 803F00 <1> cmp byte [edi], 0 ; is it free space ?
2400 0000CCB9 7639 <1> jna short systimer_7 ; yes
2401 0000CCBB FECE <1> dec bh
2402 0000CCBD 7405 <1> jz short systimer_2
2403 0000CCBF 83C710 <1> add edi, 16
2404 0000CCC2 EBF0 <1> jmp short systimer_1 ; next event space
2405 <1>
2406 <1> systimer_2:
2407 0000CCC4 58 <1> pop eax ; (*) discard
2408 <1> systimer_3:
2409 0000CCC5 C605[AC4E0100]00 <1> mov byte [u.r0], 0
2410 <1> systimer_4:
2411 0000CCCC C705[104F0100]1B00- <1> mov dword [u.error], ERR_MISC
2411 0000CCD4 0000 <1>
2412 <1> ; one of miscellaneous/other errors
2413 0000CCD6 E914F7FFFF <1> jmp error ; cf -> 1
2414 <1>
2415 <1> systimer_5:
2416 0000CCDB 883D[AC4E0100] <1> mov [u.r0], bh ; Time count unit (=2 or >3)
2417 0000CCE1 EBE9 <1> jmp short systimer_4 ; 07/06/2016
2418 <1>
2419 <1> systimer_6:
2420 0000CCE3 80FF04 <1> cmp bh, 4
2421 0000CCE6 77F3 <1> ja short systimer_5 ; undefined time count unit
2422 <1> ;jb short systimer_16
2423 <1>
2424 <1> ;mov al, 1 ; default (use current timer unit)
2425 <1> ; countdown value is in ECX !
2426 <1> ; max. value of ecx = 4294967296/10
2427 <1> ;jmp short systimer_0
2428 <1> ;jmp short systimer_19
2429 0000CCE8 74B6 <1> je short systimer_19
2430 <1>
2431 <1> systimer_16:
2432 <1> ; bh = 3
2433 <1> ; timer event via real time clock interrupt
2434 <1> ; interrupt/update frequency: 1 Hz (1 tick per second)
2435 <1>
2436 0000CCEA B0B6 <1> mov al, 182 ; (*) ; 18.2 * 10
2437 0000CCEC FE05[F54A0100] <1> inc byte [trtc] ; timer event via real time clock
2438 0000CCF2 EBAE <1> jmp short systimer_0
2439 <1>
2440 <1> systimer_7:
2441 0000CCF4 A2[AC4E0100] <1> mov [u.r0], al ; timer event number
2442 <1> ;
2443 <1> ; edi = address of empty timer event area
2444 0000CCF9 A0[FB4E0100] <1> mov al, [u.uno]
2445 0000CCFE FA <1> cli ; disable interrupts
2446 0000CCFF AA <1> stosb ; process number
2447 0000CD00 A0[F44A0100] <1> mov al, [tcallback] ; timer callback flag
2448 0000CD05 AA <1> stosb ; 1= callback method, 0= signal response byte method
2449 0000CD06 A0[F54A0100] <1> mov al, [trtc] ; timer interrupt type
2450 0000CD0B AA <1> stosb ; 1= real time clock, 0= programmable interval timer
2451 0000CD0C 88D8 <1> mov al, bl ; Signal return (Response) value
2452 0000CD0E AA <1> stosb ; response byte
2453 0000CD0F 58 <1> pop eax ; (*) ; 10 or 182
2454 0000CD10 89D3 <1> mov ebx, edx ; virtual address for response/signal byte
2455 0000CD12 F7E1 <1> mul ecx
2456 <1> ; (eax = 10 * count of 18.2 Hz timer ticks)
2457 <1> ; (count down step = 10)
2458 0000CD14 AB <1> stosd ; count limit (reset value)
2459 0000CD15 AB <1> stosd ; current count value
2460 <1>
2461 <1> ; 19/12/2016
2462 0000CD16 803D[F44A0100]00 <1> cmp byte [tcallback], 0 ; timer callback method ?
2463 0000CD1D 7604 <1> jna short systimer_17 ; no
2464 0000CD1F 89D8 <1> mov eax, ebx ; virtual address for callback routine
2465 0000CD21 EB0D <1> jmp short systimer_18
2466 <1>
2467 <1> systimer_17: ; signal response byte method
2468 <1> ; ebx = virtual address
2469 <1> ; [u.pgdir] = page directory's physical address
2470 <1> ; 20/02/2017
2471 0000CD23 FE05[F64A0100] <1> inc byte [no_page_swap] ; 1
2472 <1> ; Do not add this page to swap queue
2473 <1> ; and remove it from swap queue if it is
2474 <1> ; on the queue.
2475 0000CD29 E86085FFFF <1> call get_physical_addr
2476 0000CD2E 721A <1> jc short systimer_8 ; 07/06/2016
2477 <1> ; eax = physical address of the virtual address in user's space
2478 <1> systimer_18:
2479 0000CD30 AB <1> stosd ; response addr (physical) or callback addr (virtual)
2480 0000CD31 FE05[B7450100] <1> inc byte [timer_events] ; 07/06/201
2481 <1> ; 02/01/2017
2482 0000CD37 0FB605[FB4E0100] <1> movzx eax, byte [u.uno]

```



```

2483 0000CD3E FE80[454C0100] <1> inc byte [eax+p.timer-1]
2484 <1> ;
2485 0000CD44 FB <1> sti ; enable interrupts
2486 0000CD45 E9C5F6FFFF <1> jmp sysret
2487 <1>
2488 <1> systimer_8:
2489 <1> ; 10/06/2016
2490 <1> ; 07/06/2016
2491 0000CD4A 28C0 <1> sub al, al ; 0
2492 0000CD4C 8847F4 <1> mov [edi-12], al ; clear process number (free timer event)
2493 <1> ;mov dword [edi], eax ; 0
2494 0000CD4F FB <1> sti
2495 0000CD50 A2[AC4E0100] <1> mov [u.r0], al ; 0
2496 0000CD55 E995F6FFFF <1> jmp error
2497 <1>
2498 <1> systimer_9:
2499 <1> ; 10/06/2016
2500 <1> ; 07/06/2016
2501 0000CD5A 28C0 <1> sub al, al
2502 0000CD5C A2[AC4E0100] <1> mov byte [u.r0], al ; 0
2503 0000CD61 3805[B7450100] <1> cmp byte [timer_events], al ; 0
2504 0000CD67 7631 <1> jna short systimer_12
2505 <1>
2506 <1> ; Note: ecx and edx are undefined here
2507 <1> ; (for stop timer function)
2508 <1>
2509 0000CD69 BE[A84F0100] <1> mov esi, timer_set ; beginning address of timer events
2510 <1> ; setting space
2511 0000CD6E A0[FB4E0100] <1> mov al, [u.uno]
2512 <1>
2513 0000CD73 B710 <1> mov bh, 16
2514 <1>
2515 0000CD75 08DB <1> or bl, bl
2516 0000CD77 7544 <1> jnz short systimer_15
2517 <1>
2518 <1> ; clear timer event areas belong to current process
2519 <1> ; (for stopping all timer events belong to current process)
2520 0000CD79 FA <1> cli ; disable interrupts
2521 <1> systimer_10:
2522 <1> ; 10/06/2016
2523 <1> ; 07/06/2016
2524 0000CD7A 8A26 <1> mov ah, [esi]
2525 0000CD7C 08E4 <1> or ah, ah ; 0 ?
2526 0000CD7E 7411 <1> jz short systimer_11
2527 0000CD80 38C4 <1> cmp ah, al ; is the process number (owner) same ?
2528 0000CD82 750D <1> jne short systimer_11 ; no
2529 <1>
2530 <1> ;mov byte [esi], 0
2531 0000CD84 66C7060000 <1> mov word [esi], 0 ; clear
2532 <1> ;mov dword [esi+12], 0 ; clear
2533 <1>
2534 0000CD89 FE0D[B7450100] <1> dec byte [timer_events]
2535 0000CD8F 7409 <1> jz short systimer_12
2536 <1>
2537 <1> systimer_11:
2538 0000CD91 FECF <1> dec bh
2539 0000CD93 7405 <1> jz short systimer_12
2540 0000CD95 83C610 <1> add esi, 16
2541 0000CD98 EBE0 <1> jmp short systimer_10
2542 <1>
2543 <1> systimer_12:
2544 0000CD9A 0FB635[FB4E0100] <1> movzx esi, byte [u.uno]
2545 0000CDA1 08DB <1> or bl, bl ; all timer events or one timer event ?
2546 0000CDA3 740C <1> jz short systimer_13
2547 0000CDA5 8A9E[454C0100] <1> mov bl, [esi+p.timer-1]
2548 0000CDAB 20DB <1> and bl, bl ; previous number of timer events for the process
2549 0000CDAD 7408 <1> jz short systimer_14
2550 0000CDAF FECB <1> dec bl ; previous number of timer events for the process - 1
2551 <1> systimer_13:
2552 0000CDB1 889E[454C0100] <1> mov [esi+p.timer-1], bl ; 0 ; no timer events for process
2553 <1> systimer_14:
2554 0000CDB7 FB <1> sti ; enable interrupts
2555 0000CDB8 E952F6FFFF <1> jmp sysret
2556 <1>
2557 <1> systimer_15:
2558 0000CDBD 38FB <1> cmp bl, bh ; 16
2559 0000CDBF 0F8707FFFFFF <1> ja systimer_4 ; max. 16 timer events !
2560 <1> ;
2561 0000CDC5 88DA <1> mov dl, bl
2562 0000CDC7 FECA <1> dec dl ; 16 -> 15 ... 1 -> 0
2563 0000CDC9 C0E204 <1> shl dl, 4 ; * 16
2564 0000CDCC 0FB6FA <1> movzx edi, dl
2565 0000CDCF 01F7 <1> add edi, esi ; timer_set
2566 <1>
2567 0000CDD1 3A07 <1> cmp al, [edi] ; process number
2568 0000CDD3 0F85F3FEFFFF <1> jne systimer_4
2569 <1>
2570 <1> ; same process ID
2571 0000CDD9 FA <1> cli ; disable interrupts
2572 <1> ; 10/06/2016 ; 02/01/2017
2573 <1> ;mov byte [edi], 0
2574 0000CDDA 66C7070000 <1> mov word [edi], 0 ; clear
2575 <1> ;mov dword [edi+12], 0 ; clear
2576 0000CDDF FE0D[B7450100] <1> dec byte [timer_events]
2577 0000CDE5 EBB3 <1> jmp short systimer_12
2578 <1>
2579 <1> sysmdate: ; < change the modification time of a file >
2580 <1> ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
2581 <1> ; temporary !
2582 0000CDE7 B801000000 <1> mov eax, ERR_INV_FNUMBER ; 'invalid function number !'
2583 0000CDEC A3[104F0100] <1> mov [u.error], eax
2584 0000CDF1 A3[AC4E0100] <1> mov [u.r0], eax

```

```

2585 0000CDF6 E9F4F5FFFF      <1>      jmp      error
2586                          <1>
2587      <1> sysvideo: ; VIDEO DATA TRANSFER FUNCTIONS
2588      <1>      ; 11/07/2016
2589      <1>      ; 13/06/2016
2590      <1>      ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
2591      <1>      ;
2592      <1>      ;
2593      <1>      ; VIDEO DATA TRANSFER FUNCTIONS:
2594      <1>      ;
2595      <1>      ; Inputs:
2596      <1>      ;      BH = 0 = VIDEO BIOS Mode 3, tty/text mode data transfers
2597      <1>      ;      BL =
2598      <1>      ;          Bits 0&1, Transfer direction
2599      <1>      ;              0 - System to system
2600      <1>      ;              1 - User to system
2601      <1>      ;              2 - System to user
2602      <1>      ;              3 - User to user
2603      <1>      ;          Bits 2&3, Transfer Type
2604      <1>      ;              0 - Display page transfer
2605      <1>      ;              1 - Display page window transfer
2606      <1>      ;              2 - Frame/Viewport/Window address transfer
2607      <1>      ;              3 - Window handle transfer
2608      <1>      ;
2609      <1>      ;      /// BL = 0 -> System to system (display page) transfer
2610      <1>      ;          CL = Source page
2611      <1>      ;          DL = Destination page
2612      <1>      ;      /// BL = 1&2 -> user to system & system to user transfer
2613      <1>      ;          ECX = User buffer
2614      <1>      ;          DL = Video page
2615      <1>      ;      /// BL = 5&6 -> user to system, system to user transfer
2616      <1>      ;      (window in current display page and in current mode)
2617      <1>      ;          ESI = User's buffer address
2618      <1>      ;          ECX Low 16 bits = Top left column (X1 position)
2619      <1>      ;          ECX High 16 bits = Top row (Y1 position)
2620      <1>      ;          EDX Low 16 bits = Bottom right column (X2 position)
2621      <1>      ;          EDX High 16 bits = Bottom row (Y2 position)
2622      <1>      ;      ;
2623      <1>      ;          If BL = 5 ->
2624      <1>      ;          EDI = Swap address (in user's memory space)
2625      <1>      ;          (If swap address > 0, previous content of the window
2626      <1>      ;          will be saved into swap area in user's memory space)
2627      <1>      ;      /// BL = 4 -> system to system transfer
2628      <1>      ;          ESI = System's source buffer (video page) address
2629      <1>      ;          ECX Low 16 bits = Top left column (X1 position)
2630      <1>      ;          ECX High 16 bits = Top row (Y1 position)
2631      <1>      ;          EDX Low 16 bits = Bottom right column (X2 position)
2632      <1>      ;          EDX High 16 bits = Bottom row (Y2 position)
2633      <1>      ;          EDI = System's destination buffer (video page) address
2634      <1>      ;
2635      <1>      ;      BH = 1 = CGA Graphics (0B8000h) data transfers
2636      <1>      ;      BL =
2637      <1>      ;          0 = Fill color (color in CL] (32K)
2638      <1>      ;          1 = User to system display page transfer
2639      <1>      ;          2 = System to user display page transfer
2640      <1>      ;          3 = NOT bits in window (ECX, EDX)
2641      <1>      ;          4 = Window copy (system to system)
2642      <1>      ;          5 = User to system window transfer
2643      <1>      ;          6 = System to user window transfer
2644      <1>      ;          7 = AND display page bytes with CL
2645      <1>      ;          8 = OR display page bytes with CL
2646      <1>      ;          9 = XOR display page bytes with CL
2647      <1>      ;
2648      <1>      ;      /// BL = 0 -> Fill color (all screen pixels)
2649      <1>      ;          CL = Color value
2650      <1>      ;      /// BL = 1&2 -> user to system & system to user transfer
2651      <1>      ;          ECX = User buffer
2652      <1>      ;      /// BL = 5&6 -> user to system, system to user transfer
2653      <1>      ;      (window in current display page and in current mode)
2654      <1>      ;          ESI = User's buffer address
2655      <1>      ;          ECX Low 16 bits = Top left column (X1 position)
2656      <1>      ;          ECX High 16 bits = Top row (Y1 position)
2657      <1>      ;          EDX Low 16 bits = Bottom right column (X2 position)
2658      <1>      ;          EDX High 16 bits = Bottom row (Y2 position)
2659      <1>      ;      ;
2660      <1>      ;      /// BL = 4 -> system to system (window) transfer
2661      <1>      ;          ESI = System's source buffer (video page) address
2662      <1>      ;          ECX Low 16 bits = Top left column (X1 position)
2663      <1>      ;          ECX High 16 bits = Top row (Y1 position)
2664      <1>      ;          EDX Low 16 bits = Bottom right column (X2 position)
2665      <1>      ;          EDX High 16 bits = Bottom row (Y2 position)
2666      <1>      ;          EDI = System's destination buffer (video page) address
2667      <1>      ;      /// BL = 3 -> NOT byte in display page/memory
2668      <1>      ;          ECX Low 16 bits = Top left column (X1 position)
2669      <1>      ;          ECX High 16 bits = Top row (Y1 position)
2670      <1>      ;          EDX Low 16 bits = Bottom right column (X2 position)
2671      <1>      ;          EDX High 16 bits = Bottom row (Y2 position)
2672      <1>      ;
2673      <1>      ;      BH = 2 = VGA Graphics (0A0000h) data transfers
2674      <1>      ;      BL =
2675      <1>      ;          x0h = Fill color (color in CL] (64K)
2676      <1>      ;          x1h = User to system display page transfer
2677      <1>      ;          x2h = System to user display page transfer
2678      <1>      ;          x3h = NOT bits in window (ECX, EDX)
2679      <1>      ;          x4h = Window copy (system to system)
2680      <1>      ;          x5h = User to system window transfer
2681      <1>      ;          x6h = System to user window transfer
2682      <1>      ;          x7h = AND display page bytes with CL
2683      <1>      ;          x8h = OR display page bytes with CL
2684      <1>      ;          x9h = XOR display page bytes with CL
2685      <1>      ;          x = 0 -> screen width = 320
2686      <1>      ;          x = 1 -> screen width = 640
2687      <1>      ;          x = 2 -> screen width = 800

```

```

2687 <1> ; // BL = 0 -> Fill color (all screen pixels)
2688 <1> ; CL = Color value
2689 <1> ; // BL = 1&2 -> user to system & system to user transfer
2690 <1> ; ECX = User buffer
2691 <1> ; // BL = 5&6 -> user to system, system to user transfer
2692 <1> ; (window in current display page and in current mode)
2693 <1> ; ESI = User's buffer address
2694 <1> ; ECX Low 16 bits = Top left column (X1 position)
2695 <1> ; ECX High 16 bits = Top row (Y1 position)
2696 <1> ; EDX Low 16 bits = Bottom right column (X2 position)
2697 <1> ; EDX High 16 bits = Bottom row (Y2 position)
2698 <1> ; // BL = 4 -> system to system (window) transfer
2699 <1> ; ESI = System's source buffer (video page) address
2700 <1> ; ECX Low 16 bits = Top left column (X1 position)
2701 <1> ; ECX High 16 bits = Top row (Y1 position)
2702 <1> ; EDX Low 16 bits = Bottom right column (X2 position)
2703 <1> ; EDX High 16 bits = Bottom row (Y2 position)
2704 <1> ; EDI = System's destination buffer (video page) address
2705 <1> ; // BL = 3 -> NOT byte in display page/memory
2706 <1> ; ECX Low 16 bits = Top left column (X1 position)
2707 <1> ; ECX High 16 bits = Top row (Y1 position)
2708 <1> ; EDX Low 16 bits = Bottom right column (X2 position)
2709 <1> ; EDX High 16 bits = Bottom row (Y2 position)
2710 <1> ;
2711 <1> ; BH = 3 = Super VGA, LINEAR FRAME BUFFER data transfers
2712 <1> ; BL =
2713 <1> ; 0 = Fill color (color in ECX] (Frame buffer size)
2714 <1> ; 1 = User to system display page transfer
2715 <1> ; 2 = System to user display page transfer
2716 <1> ; 3 = NOT bits in window (ECX, EDX)
2717 <1> ; 4 = Window copy (system to system)
2718 <1> ; 5 = User to system window transfer
2719 <1> ; 6 = System to user window transfer
2720 <1> ; 7 = AND display page bytes with ECX
2721 <1> ; 8 = OR display page bytes with ECX
2722 <1> ; 9 = XOR display page bytes with ECX
2723 <1> ;
2724 <1> ; // BL = 0 -> Fill color (all screen pixels)
2725 <1> ; CL = Color value
2726 <1> ; // BL = 1&2 -> user to system & system to user transfer
2727 <1> ; ECX = User buffer
2728 <1> ; // BL = 5&6 -> user to system, system to user transfer
2729 <1> ; (window in current display page and in current mode)
2730 <1> ; ESI = User's buffer address
2731 <1> ; ECX Low 16 bits = Top left column (X1 position)
2732 <1> ; ECX High 16 bits = Top row (Y1 position)
2733 <1> ; EDX Low 16 bits = Bottom right column (X2 position)
2734 <1> ; EDX High 16 bits = Bottom row (Y2 position)
2735 <1> ; // BL = 4 -> system to system (window) transfer
2736 <1> ; ESI = System's source buffer (video page) address
2737 <1> ; ECX Low 16 bits = Top left column (X1 position)
2738 <1> ; ECX High 16 bits = Top row (Y1 position)
2739 <1> ; EDX Low 16 bits = Bottom right column (X2 position)
2740 <1> ; EDX High 16 bits = Bottom row (Y2 position)
2741 <1> ; EDI = System's destination buffer (video page) address
2742 <1> ; // BL = 3 -> NOT byte in display page/memory
2743 <1> ; ECX Low 16 bits = Top left column (X1 position)
2744 <1> ; ECX High 16 bits = Top row (Y1 position)
2745 <1> ; EDX Low 16 bits = Bottom right column (X2 position)
2746 <1> ; EDX High 16 bits = Bottom row (Y2 position)
2747 <1> ;
2748 <1> ; Outputs:
2749 <1> ; EAX = transfer/byte count
2750 <1> ;
2751 <1> ; NOTE: If the source or destination address passes out of
2752 <1> ; video pages (display memory limits), data will not be transferred
2753 <1> ; and EAX will return as 0.
2754 <1> ;
2755 <1> ;
2756 <1> ; DIRECT (STANDARD VGA/CGA) DISPLAY MEMORY ACCESS FUNCTIONS:
2757 <1> ;
2758 <1> ; BH = 4 = CGA direct video memory (0B8000h, 32K) access
2759 <1> ; Page directory & page tables of the user's
2760 <1> ; program will be updated to direct access to
2761 <1> ; 0B8000h (32K) video (CGA, color) memory; if
2762 <1> ; there is not a permission conflict or lock!
2763 <1> ; (User's program/process will have permission to
2764 <1> ; access locked display memory if the owner is
2765 <1> ; it's parent.)
2766 <1> ;
2767 <1> ; Screen width = 320
2768 <1> ;
2769 <1> ; BH = 5 = VGA direct video memory (0A0000h, 64K) access
2770 <1> ; Page directory & page tables of the user's
2771 <1> ; program will be updated to direct access to
2772 <1> ; 0A0000h (64K) video (VGA) memory; if there is not
2773 <1> ; a permission conflict or lock!
2774 <1> ; (User's program/process will have permission to
2775 <1> ; access locked display memory if the owner is
2776 <1> ; it's parent.)
2777 <1> ;
2778 <1> ; BL = Screen width (320, 640, 800)
2779 <1> ;
2780 <1> ; Outputs:
2781 <1> ; EAX = Display memory address for direct access
2782 <1> ; 0A0000h for VGA, 0B8000h for CGA
2783 <1> ; (Display memory size: 32K for CGA, 64K for VGA)
2784 <1> ; EAX = 0 if display page access permission has been denied.
2785 <1> ; (Locked!)
2786 <1> ;
2787 <1> ; LINEAR FRAME BUFFER ACCESS FUNCTIONS:
2788 <1> ;

```

```

2789      <1>      ;      BH = 6 = Linear Frame Buffer direct video memory access
2790      <1>      ;
2791      <1>      ;      Page directory & page tables of the user's
2792      <1>      ;      program will be updated to direct access to
2793      <1>      ;      the configured LFB (Linear Frame Buffer) address,
2794      <1>      ;      if there is not a permission conflict or lock!
2795      <1>      ;      (User's program/process will have permission to
2796      <1>      ;      access locked display memory if the owner is
2797      <1>      ;      it's parent.)
2798      <1>      ;
2799      <1>      ;      Return: EAX = Linear Frame Buffer address
2800      <1>      ;      EDX = Frame Buffer Size in bytes
2801      <1>      ;
2802      <1>      ;      BH = 7 = Get Linear Frame Buffer info (for current mode)
2803      <1>      ;
2804      <1>      ;      Return:
2805      <1>      ;      EAX = Frame Buffer Address (0 = is not in use)
2806      <1>      ;      EDX = Frame Buffer Size in bytes
2807      <1>      ;      BL = Current Video Mode
2808      <1>      ;      BL = 0FFh -> Super VGA (Extended VGA)
2809      <1>      ;      If BL = 0FFh,
2810      <1>      ;      ;      BH = 0 = 16 colors
2811      <1>      ;      ;      BH = 1 = 256 colors
2812      <1>      ;      ;      BH = 2 = 66536 colors
2813      <1>      ;      ;      BH = 3 = 24 bits TRUE (16M) colors
2814      <1>      ;      ;      BH = 4 = 32 bits TRUE (16M) colors
2815      <1>      ;      ECX = Pixel resolution
2816      <1>      ;      CX = Width (640, 800, 1024, 1366, 1920)
2817      <1>      ;      High 16 bits of ECX = Height
2818      <1>      ;
2819      <1>      ;      NOTE: Each process will have it's own frame buffer
2820      <1>      ;      address and resolution parameters in 'u' area.
2821      <1>      ;      Then, if the current frame buffer & resolution
2822      <1>      ;      is different, frame buffer r/w functions
2823      <1>      ;      will use scale factor to convert process's
2824      <1>      ;      pixel coordinates to actual screen coordinates.
2825      <1>      ;      resolution -> dimensional scale
2826      <1>      ;      color size -> color scale
2827      <1>      ;      * RGB (TRUE) colors to 256 colors conversion:
2828      <1>      ;      TRUE Colors -> 8,8,8 (R,G,B; byte 0 is R)
2829      <1>      ;      256 colors -> 2,2,2,2 (R,G,B,L; bit 0&1 is R)
2830      <1>      ;      bit 6&7 -> luminosity base level (0,1,2,3)
2831      <1>      ;      bit 4&5 -> blue level (0,1,2,3)
2832      <1>      ;      bit 2&3 -> green level (0,1,2,3)
2833      <1>      ;      bit 0&1 -> red level (0,1,2,3)
2834      <1>      ;      Example: total red level : luminosity + red level
2835      <1>      ;      Luminosity base level: 0 -> 16
2836      <1>      ;      1 -> 32
2837      <1>      ;      2 -> 64
2838      <1>      ;      3 -> 128
2839      <1>      ;      Color level:
2840      <1>      ;      0 -> 0
2841      <1>      ;      1 -> luminosity level
2842      <1>      ;      2 -> luminosity level + 64
2843      <1>      ;      3 -> 255
2844      <1>      ;      Luminosity base level = min (R,G,B)
2845      <1>      ;      if it is <16, it will be set to 16
2846      <1>      ;      Color levels: Color values are fixed to (nearest)
2847      <1>      ;      one of all possible set level (step) values
2848      <1>      ;      (according to luminosity base level); then
2849      <1>      ;      color levels are set to R-L, G-L, B-L.
2850      <1>      ;      For example: If luminosity base level is 32
2851      <1>      ;      all possible set values are 0, 32, 96, 255.
2852      <1>      ;
2853      <1>      ;      * RGB (TRUE) colors to 16 colors conversion:
2854      <1>      ;      16 colors: R, B,G, L bits (4 bits)
2855      <1>      ;      If any one of R,G,B >= 128 L = 1
2856      <1>      ;      If max. value of (R,G,B) >= 32, it is 1
2857      <1>      ;      else all color bits (R&G&B&L) are 0
2858      <1>      ;      If the second value >= max. value / 2
2859      <1>      ;      it is 1
2860      <1>      ;      If third value value >= max. value / 2
2861      <1>      ;      it is 1
2862      <1>      ;      Example: R = 132, G = 64, B = 78
2863      <1>      ;      L = 1, R = 1
2864      <1>      ;      G < 66 --> G = 0
2865      <1>      ;      B >= 66 --> B = 1
2866
2867
2868      <1>      ; 16/05/2016
2868      0000CDFB 31C0      xor     eax, eax
2869      0000CDFD A3[AC4E0100] mov    [u.r0], eax
2870
2871      0000CE02 20FF      and     bh, bh
2872      0000CE04 0F8572020000 jnz    sysvideo_13 ; 11/07/2016
2873
2874      <1>      ; Video mode 0, 80*25 text mode, CGA 16 colors ; [CRT_MODE] = 3
2875      0000CE0A 88DF      mov     bh, bl
2876      0000CE0C C0EF02      shr     bh, 2
2877      0000CE0F 20FF      and     bh, bh
2878      0000CE11 0F8598000000 jnz    sysvideo_4
2879      0000CE17 BF00800B00      mov     edi, 0B8000h
2880      0000CE1C 20D2      and     dl, dl
2881      0000CE1E 7413      jz      short sysvideo_1
2882      0000CE20 80FA07      cmp     dl, 7
2883      0000CE23 0F87E6F5FFFF      ja      sysret
2884      sysvideo_0:
2885      0000CE29 81C7A00F0000      add     edi, 80*25*2
2886      0000CE2F FECA      dec     dl
2887      0000CE31 75F6      jnz     short sysvideo_0
2888      sysvideo_1:
2889      0000CE33 80E303      and     bl, 3

```



```

2890 0000CE36 7530      <1>      jnz      short sysvideo_2
2891 0000CE38 80F907     <1>      cmp      cl, 7
2892 0000CE3B 0F87CEF5FFFF    <1>      ja       sysret
2893                <1>      ; system to system video/display page transfer (mode 0)
2894 0000CE41 BE00800B00     <1>      mov      esi, 0B8000h
2895 0000CE46 0FB6C1      <1>      movzx     eax, cl
2896 0000CE49 BAA00F0000     <1>      mov      edx, 80*25*2
2897 0000CE4E F7E2      <1>      mul      edx
2898 0000CE50 01C6      <1>      add      esi, eax
2899 0000CE52 B9A00F0000     <1>      mov      ecx, (80*25*2)
2900 0000CE57 890D[AC4E0100]    <1>      mov      [u.r0], ecx
2901 0000CE5D 66C1E902     <1>      shr      cx, 2 ; /4
2902 0000CE61 F3A5      <1>      rep      movsd
2903 0000CE63 E9A7F5FFFF    <1>      jmp      sysret
2904                <1>      sysvideo_2:
2905 0000CE68 80FB02     <1>      cmp      bl, 2
2906 0000CE6B 0F879EF5FFFF    <1>      ja       sysret
2907 0000CE71 721F      <1>      jnb      short sysvideo_3
2908                <1>      ; system to user video/display page transfer (mode 0)
2909 0000CE73 89FE      <1>      mov      esi, edi
2910 0000CE75 89CF      <1>      mov      edi, ecx ; user buffer
2911 0000CE77 B9A00F0000     <1>      mov      ecx, 80*25*2
2912 0000CE7C E84F190000     <1>      call     transfer_to_user_buffer ; fast transfer
2913 0000CE81 0F8288F5FFFF    <1>      jc       sysret
2914 0000CE87 890D[AC4E0100]    <1>      mov      [u.r0], ecx
2915 0000CE8D E97DF5FFFF    <1>      jmp      sysret
2916                <1>      sysvideo_3:
2917                <1>      ; user to system video/display page transfer (mode 0)
2918 0000CE92 89CE      <1>      mov      esi, ecx ; user buffer
2919                <1>      ; edi = video page address
2920 0000CE94 B9A00F0000     <1>      mov      ecx, 80*25*2
2921 0000CE99 E87C190000     <1>      call     transfer_from_user_buffer ; fast transfer
2922 0000CE9E 0F826BF5FFFF    <1>      jc       sysret
2923 0000CEA4 890D[AC4E0100]    <1>      mov      [u.r0], ecx
2924 0000CEAA E960F5FFFF    <1>      jmp      sysret
2925                <1>      sysvideo_4:
2926 0000CEAF 80E303     <1>      and      bl, 3
2927 0000CEB2 0F85F6000000    <1>      jnz      sysvideo_9
2928 0000CEB8 80F907     <1>      cmp      cl, 7
2929 0000CEBB 0F874EF5FFFF    <1>      ja       sysret
2930                <1>      ; system to system video/display page window transfer (mode 0)
2931 0000CEC1 81FE00800B00     <1>      cmp      esi, 0B8000h
2932 0000CEC7 0F8242F5FFFF    <1>      jnb      sysret
2933 0000CECD 81FE00FD0B00     <1>      cmp      esi, 0B8000h+(80*25*2*8)
2934 0000CED3 0F8336F5FFFF    <1>      jnb      sysret
2935 0000CED9 81FF00800B00     <1>      cmp      edi, 0B8000h
2936 0000CEDF 0F822AF5FFFF    <1>      jnb      sysret
2937 0000CEE5 81FF00FD0B00     <1>      cmp      edi, 0B8000h+(80*25*2*8)
2938 0000CEEB 0F831EF5FFFF    <1>      jnb      sysret
2939                <1>      ;
2940 0000CEF1 51      <1>      push     ecx
2941 0000CEF2 52      <1>      push     edx
2942 0000CEF3 0FB7C1     <1>      movzx     eax, cx ; top left column
2943 0000CEF6 50      <1>      push     eax
2944 0000CEF7 C1E910     <1>      shr      ecx, 16 ; top row
2945 0000CEFA 66B8A000     <1>      mov      ax, 80*2 ; 80 colums, 160 bytes per row
2946 0000CEFE F7E1      <1>      mul      ecx
2947 0000CF00 01C6      <1>      add      esi, eax
2948 0000CF02 01C7      <1>      add      edi, eax
2949 0000CF04 58      <1>      pop      eax
2950 0000CF05 66D1E0     <1>      shl      ax, 1 ; *2
2951 0000CF08 01C6      <1>      add      esi, eax
2952 0000CF0A 01C7      <1>      add      edi, eax
2953 0000CF0C 5A      <1>      pop      edx
2954 0000CF0D 59      <1>      pop      ecx
2955 0000CF0E B800FD0B00     <1>      mov      eax, 0B8000h+(80*25*2*8)
2956 0000CF13 39C6      <1>      cmp      esi, eax
2957 0000CF15 0F83F4F4FFFF    <1>      jnb      sysret
2958 0000CF1B 39C6      <1>      cmp      esi, eax
2959 0000CF1D 0F83ECF4FFFF    <1>      jnb      sysret
2960                <1>
2961 0000CF23 56      <1>      push     esi ; ****
2962 0000CF24 57      <1>      push     edi ; ***
2963 0000CF25 52      <1>      push     edx ; **
2964 0000CF26 51      <1>      push     ecx ; *
2965 0000CF27 C1E910     <1>      shr      ecx, 16 ; top row
2966 0000CF2A C1EA10     <1>      shr      edx, 16 ; bottom row
2967 0000CF2D 83F918     <1>      cmp      ecx, 24 ; max. 25 rows
2968 0000CF30 7773      <1>      ja       short sysvideo_6
2969 0000CF32 83FA18     <1>      cmp      edx, 24 ; max. 25 rows
2970 0000CF35 776E      <1>      ja       short sysvideo_6
2971 0000CF37 28CA      <1>      sub      dl, cl
2972 0000CF39 726A      <1>      jc       short sysvideo_6
2973 0000CF3B 50      <1>      push     eax ; *****
2974 0000CF3C 89D3      <1>      mov      ebx, edx ; row count - 1
2975 0000CF3E B8A0000000    <1>      mov      eax, 80*2
2976 0000CF43 F7E0      <1>      mul      eax
2977 0000CF45 01C6      <1>      add      esi, eax
2978 0000CF47 01C7      <1>      add      edi, eax
2979 0000CF49 58      <1>      pop      eax ; *****
2980 0000CF4A 39C6      <1>      cmp      esi, eax
2981 0000CF4C 7757      <1>      ja       short sysvideo_6
2982 0000CF4E 39C7      <1>      cmp      edi, eax
2983 0000CF50 7753      <1>      ja       short sysvideo_6
2984 0000CF52 59      <1>      pop      ecx ; *
2985 0000CF53 5A      <1>      pop      edx ; **
2986 0000CF54 81E1FFFF0000 <1>      and      ecx, 0FFFFh
2987 0000CF5A 81E2FFFF0000 <1>      and      edx, 0FFFFh
2988 0000CF60 83F94F     <1>      cmp      ecx, 79 ; max. 80 columns
2989 0000CF63 7742      <1>      ja       short sysvideo_7
2990 0000CF65 83FA4F     <1>      cmp      edx, 79 ; max. 80 columns
2991 0000CF68 773D      <1>      ja       short sysvideo_7

```

```

2992 0000CF6A 28CA      <1>      sub    dl, cl
2993 0000CF6C 7639      <1>      jna    short sysvideo_7
2994                   <1>      ; edx = column count (width) - 1
2995 0000CF6E D0E2      <1>      shl    dl, 1
2996 0000CF70 01D6      <1>      add    esi, edx
2997 0000CF72 01D7      <1>      add    edi, edx
2998 0000CF74 39C6      <1>      cmp    esi, eax
2999 0000CF76 772F      <1>      ja     short sysvideo_7
3000 0000CF78 39C7      <1>      cmp    edi, eax
3001 0000CF7A 772B      <1>      ja     short sysvideo_7
3002 0000CF7C 5F          <1>      pop    edi ; ***
3003 0000CF7D 5E          <1>      pop    esi ; ****
3004 0000CF7E FEC3      <1>      inc    bl
3005 0000CF80 FEC2      <1>      inc    dl ; column count
3006 0000CF82 88D7      <1>      mov    bh, dl
3007 0000CF84 D0E2      <1>      shl    dl, 1
3008 0000CF86 B8A0000000      <1>      mov    eax, 80*2
3009 0000CF8B 28D0      <1>      sub    al, dl ; (80 - columns) * 2
3010                   <1> sysvideo_5:
3011 0000CF8D 88F9      <1>      mov    cl, bh
3012 0000CF8F 0115[AC4E0100] <1>      add    [u.r0], edx
3013 0000CF95 F366A5      <1>      rep    movsw
3014 0000CF98 01C6      <1>      add    esi, eax ; next row
3015 0000CF9A 01C7      <1>      add    edi, eax ; next row
3016 0000CF9C FECB      <1>      dec    bl
3017 0000CF9E 75ED      <1>      jnz    short sysvideo_5
3018 0000CFA0 E96AF4FFFF      <1>      jmp    sysret
3019                   <1>
3020                   <1> sysvideo_6:
3021 0000CFA5 59          <1>      pop    ecx ; *
3022 0000CFA6 5A          <1>      pop    edx ; **
3023                   <1> sysvideo_7:
3024 0000CFA7 5F          <1>      pop    edi ; ***
3025 0000CFA8 5E          <1>      pop    esi ; ****
3026 0000CFA9 E961F4FFFF      <1>      jmp    sysret
3027                   <1>
3028                   <1> sysvideo_9:
3029 0000CFAE 80FB02      <1>      cmp    bl, 2
3030 0000CFB1 0F8758F4FFFF      <1>      ja     sysret
3031                   <1>
3032 0000CFB7 56          <1>      push   esi ; ****
3033 0000CFB8 57          <1>      push   edi ; ***
3034 0000CFB9 52          <1>      push   edx ; **
3035 0000CFBA 51          <1>      push   ecx ; *
3036                   <1>
3037 0000CFBB C1E910      <1>      shr    ecx, 16 ; top row
3038 0000CFBE C1EA10      <1>      shr    edx, 16 ; bottom row
3039 0000CFC1 83F918      <1>      cmp    ecx, 24 ; max. 25 rows
3040 0000CFC4 77DF      <1>      ja     short sysvideo_6
3041 0000CFC6 83FA18      <1>      cmp    edx, 24 ; max. 25 rows
3042 0000CFC9 77DA      <1>      ja     short sysvideo_6
3043 0000CFCB 28CA      <1>      sub    dl, cl
3044 0000CFCD 72D6      <1>      jc     short sysvideo_6
3045                   <1>
3046 0000CFCF 88CD      <1>      mov    ch, cl ; top row
3047 0000CFD1 8A0D[4E380100] <1>      mov    cl, [ACTIVE_PAGE]
3048 0000CFD7 BFA00F0000      <1>      mov    edi, 80*25*2
3049 0000CFDC D3E7      <1>      shl    edi, cl
3050 0000CFDE 81C760700B00      <1>      add    edi, 0B8000h - 80*25*2
3051                   <1>
3052 0000CFE4 88D7      <1>      mov    bh, dl ; row count - 1
3053 0000CFE6 88EA      <1>      mov    dl, ch ; top row
3054 0000CFE8 B8A0000000      <1>      mov    eax, 80*2
3055 0000CFED F7E2      <1>      mul    edx
3056 0000CFEF 01C7      <1>      add    edi, eax
3057                   <1>
3058 0000CFF1 59          <1>      pop    ecx ; *
3059 0000CFF2 5A          <1>      pop    edx ; **
3060 0000CFF3 81E1FFFF0000      <1>      and    ecx, 0FFFFh
3061 0000CFF9 81E2FFFF0000      <1>      and    edx, 0FFFFh
3062 0000CFFF 83F94F      <1>      cmp    ecx, 79 ; max. 80 columns
3063 0000D002 77A3      <1>      ja     short sysvideo_7
3064 0000D004 83FA4F      <1>      cmp    edx, 79 ; max. 80 columns
3065 0000D007 779E      <1>      ja     short sysvideo_7
3066                   <1>
3067 0000D009 28CA      <1>      sub    dl, cl
3068 0000D00B 769A      <1>      jna    short sysvideo_7
3069                   <1>
3070 0000D00D 0FB6C1      <1>      movzx   eax, cl ; left column
3071 0000D010 D0E0      <1>      shl    al, 1 ; column * 2
3072 0000D012 01C7      <1>      add    edi, eax
3073                   <1>
3074 0000D014 FEC2      <1>      inc    dl ; column count
3075 0000D016 D0E2      <1>      shl    dl, 1
3076 0000D018 88D1      <1>      mov    cl, dl ; column count * 2
3077 0000D01A B2A0      <1>      mov    dl, 80*2
3078 0000D01C 58          <1>      pop    eax ; *** (swap address)
3079 0000D01D 5E          <1>      pop    esi ; ****
3080 0000D01E FEC7      <1>      inc    bh
3081                   <1>
3082                   <1> ;mov    edx, 80*2
3083 0000D020 B2A0      <1>      mov    dl, 80*2
3084                   <1> ;
3085 0000D022 80FB01      <1>      cmp    bl, 1
3086 0000D025 7735      <1>      ja     short sysvideo_11
3087                   <1>
3088                   <1> ; user to system video/display page window transfer (mode 0)
3089 0000D027 21C0      <1>      and    eax, eax ; swap address
3090 0000D029 7413      <1>      jz     short sysvideo_10 ; no window swap
3091                   <1> ; save previous window content in user's buffer (swap address)
3092 0000D02B 56          <1>      push   esi ; user buffer
3093 0000D02C 57          <1>      push   edi ; beginning address of the window

```

```

3094 0000D02D 89FE      <1>      mov     esi, edi
3095 0000D02F 89C7      <1>      mov     edi, eax
3096 0000D031 E89A170000    <1>      call    transfer_to_user_buffer ; fast transfer
3097 0000D036 5F        <1>      pop     edi
3098 0000D037 5E        <1>      pop     esi
3099 0000D038 0F82D1F3FFFF    <1>      jc      sysret
3100                      <1> sysvideo_10:
3101                      <1>      ; user to system video/display page window transfer (mode 0)
3102                      <1>      ; esi =      user buffer
3103 0000D03E E8D7170000    <1>      call    transfer_from_user_buffer ; fast transfer
3104 0000D043 0F82C6F3FFFF    <1>      jc      sysret
3105 0000D049 010D[AC4E0100] <1>      add     [u.r0], ecx
3106 0000D04F 01D7      <1>      add     edi, edx ; next row
3107 0000D051 01CE      <1>      add     esi, ecx
3108 0000D053 FECF      <1>      dec     bh
3109 0000D055 75E7      <1>      jnz     short sysvideo_10
3110 0000D057 E9B3F3FFFF    <1>      jmp     sysret
3111                      <1>
3112                      <1> sysvideo_11:
3113                      <1>      ; system to user video/display page window transfer (mode 0)
3114 0000D05C 87FE      <1>      xchg    edi, esi
3115                      <1> sysvideo_12:
3116                      <1>      ; esi = beginning address of the window
3117                      <1>      ; edi =      user buffer
3118 0000D05E E86D170000    <1>      call    transfer_to_user_buffer ; fast transfer
3119 0000D063 0F82A6F3FFFF    <1>      jc      sysret
3120 0000D069 010D[AC4E0100] <1>      add     [u.r0], ecx
3121 0000D06F 01D6      <1>      add     esi, edx ; next row
3122 0000D071 01CF      <1>      add     edi, ecx
3123 0000D073 FECF      <1>      dec     bh
3124 0000D075 75E7      <1>      jnz     short sysvideo_12
3125 0000D077 E993F3FFFF    <1>      jmp     sysret
3126                      <1>
3127                      <1> sysvideo_13:
3128 0000D07C 80FF01      <1>      cmp     bh, 1
3129 0000D07F 0F871F030000    <1>      ja      sysvideo_38
3130                      <1>      ; BH = 1 = CGA Graphics (0B8000h) data transfers
3131                      <1>
3132 0000D085 20DB      <1>      and     bl, bl
3133 0000D087 751A      <1>      jnz     short sysvideo_14
3134                      <1>
3135                      <1>      ; BL = 0 = Fill color (color in CL) (32K)
3136                      <1>
3137 0000D089 88C8      <1>      mov     al, cl
3138 0000D08B B900800000    <1>      mov     ecx, 32768
3139 0000D090 66890D[AC4E0100] <1>      mov     [u.r0], cx
3140 0000D097 BF00800B00    <1>      mov     edi, 0B8000h
3141 0000D09C F3AB      <1>      rep     stosd
3142 0000D09E E96CF3FFFF    <1>      jmp     sysret
3143                      <1>
3144                      <1> sysvideo_14:
3145 0000D0A3 80FB01      <1>      cmp     bl, 1
3146 0000D0A6 7723      <1>      ja      short sysvideo_16
3147                      <1>
3148 0000D0A8 89CE      <1>      mov     esi, ecx ; user buffer
3149                      <1>      ; BL = 1 = user to system video/display page transfer
3150                      <1> sysvideo_15:
3151 0000D0AA BF00800B00    <1>      mov     edi, 0B8000h
3152                      <1>      ; edi = video page address
3153 0000D0AF B900800000    <1>      mov     ecx, 32768
3154 0000D0B4 E861170000    <1>      call    transfer_from_user_buffer ; fast transfer
3155 0000D0B9 0F8250F3FFFF    <1>      jc      sysret ; [u.r0] = 0
3156 0000D0BF 66890D[AC4E0100] <1>      mov     [u.r0], cx
3157 0000D0C6 E944F3FFFF    <1>      jmp     sysret
3158                      <1>
3159                      <1> sysvideo_16:
3160 0000D0CB 80FB02      <1>      cmp     bl, 2
3161 0000D0CE 7723      <1>      ja      short sysvideo_18
3162                      <1>
3163 0000D0D0 89CF      <1>      mov     edi, ecx ; user buffer
3164                      <1>      ; BL = 2 = system to user video/display page transfer
3165                      <1> sysvideo_17:
3166 0000D0D2 BE00800B00    <1>      mov     esi, 0B8000h
3167 0000D0D7 B900800000    <1>      mov     ecx, 32768
3168 0000D0DC E8EF160000    <1>      call    transfer_to_user_buffer ; fast transfer
3169 0000D0E1 0F8228F3FFFF    <1>      jc      sysret ; [u.r0] = 0
3170 0000D0E7 66890D[AC4E0100] <1>      mov     [u.r0], cx
3171 0000D0EE E91CF3FFFF    <1>      jmp     sysret
3172                      <1>
3173                      <1> sysvideo_18:
3174 0000D0F3 80FB03      <1>      cmp     bl, 3
3175 0000D0F6 777E      <1>      ja      short sysvideo_23
3176                      <1>
3177                      <1>      ; BL = 3 = NOT bits in window (ECX, EDX)
3178                      <1>
3179 0000D0F8 BF00800B00    <1>      mov     edi, 0B8000h
3180 0000D0FD 89FE      <1>      mov     esi, edi
3181                      <1>
3182 0000D0FF 39CA      <1>      cmp     edx, ecx ; bottom-right > top-left ?
3183 0000D101 7716      <1>      ja      short sysvideo_20 ; window
3184                      <1>      ; full screen (update)
3185 0000D103 B900800000    <1>      mov     ecx, 32768
3186 0000D108 66890D[AC4E0100] <1>      mov     [u.r0], cx
3187                      <1> sysvideo_19:
3188 0000D10F F616      <1>      not     byte [esi] ; NOT operation
3189 0000D111 46        <1>      inc     esi
3190 0000D112 E2FB      <1>      loop    sysvideo_19
3191 0000D114 E9F6F2FFFF    <1>      jmp     sysret
3192                      <1> sysvideo_20:
3193 0000D119 0FB7C2      <1>      movzx   eax, dx ; bottom right column
3194 0000D11C 6629C8      <1>      sub     ax, cx ; - top left column
3195 0000D11F 0F82EAF2FFFF    <1>      jb      sysret ; invalid

```

```

3196 0000D125 6640      <1>      inc    ax ; same column no == 1 column
3197 0000D127 50       <1>      push   eax ; byte count per window row
3198 0000D128 52       <1>      push   edx
3199 0000D129 BB40010000 <1>      mov    ebx, 320 ; screen width
3200 0000D12E 89C8     <1>      mov    eax, ecx
3201 0000D130 C1E810   <1>      shr    eax, 16 ; top row
3202 0000D133 F7E3     <1>      mul    ebx
3203 0000D135 6689CA   <1>      mov    dx, cx ; top left column
3204 0000D138 01D0     <1>      add    eax, edx
3205 0000D13A 01C6     <1>      add    esi, eax ; start address
3206 0000D13C 59       <1>      pop    ecx ; edx
3207 0000D13D 89C8     <1>      mov    eax, ecx
3208 0000D13F C1E810   <1>      shr    eax, 16 ; bottom row
3209 0000D142 F7E3     <1>      mul    ebx
3210 0000D144 6689CA   <1>      mov    dx, cx ; bottom right column
3211 0000D147 01D0     <1>      add    eax, edx
3212 0000D149 01C7     <1>      add    edi, eax ; stop address (included)
3213 0000D14B 5A       <1>      pop    edx ; byte count per window row
3214 0000D14C 81FFFFFFF0B00 <1>      cmp    edi, 0BFFFFFFh
3215 0000D152 0F87B7F2FFFF <1>      ja     sysret
3216 0000D158 56       <1>      push   esi
3217 0000D159 4E       <1>      dec    esi
3218                                <1> sysvideo_21:
3219 0000D15A 89D1     <1>      mov    ecx, edx
3220                                <1> sysvideo_22:
3221 0000D15C 46       <1>      inc    esi
3222 0000D15D F616     <1>      not    byte [esi]
3223 0000D15F E2FB     <1>      loop   sysvideo_22
3224 0000D161 01DE     <1>      add    esi, ebx ; bytes per screen row
3225                                <1>      ;
3226 0000D163 39FE     <1>      cmp    esi, edi ; stop address (included in loop)
3227 0000D165 76F3     <1>      jna     short sysvideo_21
3228 0000D167 5E       <1>      pop    esi
3229 0000D168 29F7     <1>      sub    edi, esi
3230 0000D16A 66893D[AC4E0100] <1>      mov    [u.r0], di
3231 0000D171 E999F2FFFF <1>      jmp    sysret
3232                                <1>
3233                                <1> sysvideo_23:
3234 0000D176 80FB04   <1>      cmp    bl, 4
3235 0000D179 0F87A7000000 <1>      ja     sysvideo_26
3236                                <1>
3237                                <1>      ; BL = 4 = window copy (system to system)
3238                                <1>
3239 0000D17F B800800B00 <1>      mov    eax, 0B8000h
3240 0000D184 39C6     <1>      cmp    esi, eax
3241 0000D186 0F8283F2FFFF <1>      jb     sysret
3242 0000D18C 39C7     <1>      cmp    edi, eax
3243 0000D18E 0F827BF2FFFF <1>      jb     sysret
3244 0000D194 6605FF7F <1>      add    ax, 7FFFh ; 32767
3245 0000D198 39C6     <1>      cmp    esi, eax
3246 0000D19A 0F876FF2FFFF <1>      ja     sysret
3247 0000D1A0 39C7     <1>      cmp    edi, eax
3248 0000D1A2 0F8767F2FFFF <1>      ja     sysret
3249                                <1>
3250 0000D1A8 39CA     <1>      cmp    edx, ecx ; bottom-right > top-left ?
3251 0000D1AA 7714     <1>      ja     short sysvideo_24 ; window
3252                                <1>      ; full screen copy
3253 0000D1AC 89C1     <1>      mov    ecx, eax
3254 0000D1AE 29F9     <1>      sub    ecx, edi
3255 0000D1B0 6641     <1>      inc    cx
3256 0000D1B2 66890D[AC4E0100] <1>      mov    [u.r0], cx
3257 0000D1B9 F3A4     <1>      rep    movsb
3258 0000D1BB E94FF2FFFF <1>      jmp    sysret
3259                                <1> sysvideo_24:
3260 0000D1C0 0FB7C2   <1>      movzx   eax, dx ; bottom right column
3261 0000D1C3 6629C8   <1>      sub    ax, cx ; - top left column
3262 0000D1C6 0F8243F2FFFF <1>      jb     sysret ; invalid
3263 0000D1CC 6640     <1>      inc    ax ; same column no == 1 column
3264 0000D1CE 50       <1>      push   eax ; byte count per window row
3265                                <1>      ;
3266 0000D1CF 52       <1>      push   edx
3267 0000D1D0 BB40010000 <1>      mov    ebx, 320 ; screen width
3268 0000D1D5 89C8     <1>      mov    eax, ecx
3269 0000D1D7 C1E810   <1>      shr    eax, 16 ; top row
3270 0000D1DA F7E3     <1>      mul    ebx
3271 0000D1DC 6689CA   <1>      mov    dx, cx ; top left column
3272 0000D1DF 01D0     <1>      add    eax, edx
3273 0000D1E1 01C7     <1>      add    edi, eax ; start address
3274 0000D1E3 01C6     <1>      add    esi, eax
3275 0000D1E5 59       <1>      pop    ecx ; edx
3276 0000D1E6 89C8     <1>      mov    eax, ecx
3277 0000D1E8 C1E810   <1>      shr    eax, 16 ; bottom row
3278 0000D1EB F7E3     <1>      mul    ebx
3279 0000D1ED 6689CA   <1>      mov    dx, cx ; bottom right column
3280 0000D1F0 01D0     <1>      add    eax, edx
3281 0000D1F2 5A       <1>      pop    edx ; byte count per window row
3282 0000D1F3 0500800B00 <1>      add    eax, 0B8000h
3283 0000D1F8 3DFFFFFFF0B00 <1>      cmp    eax, 0BFFFFFFh
3284 0000D1FD 0F870CF2FFFF <1>      ja     sysret
3285 0000D203 57       <1>      push   edi ; start address
3286 0000D204 50       <1>      push   eax ; stop address (included)
3287                                <1> sysvideo_25:
3288 0000D205 89D1     <1>      mov    ecx, edx
3289 0000D207 F3A4     <1>      rep    movsb
3290 0000D209 4F       <1>      dec    edi
3291 0000D20A 4E       <1>      dec    esi
3292 0000D20B 01DF     <1>      add    edi, ebx ; bytes per screen row
3293 0000D20D 01DE     <1>      add    esi, ebx
3294                                <1>      ;
3295 0000D20F 3B3C24   <1>      cmp    edi, [esp] ; stop addr(included in loop)
3296 0000D212 76F1     <1>      jna     short sysvideo_25
3297 0000D214 5B       <1>      pop    ebx ; stop address

```



```

3298 0000D215 5F          <1>      pop     edi ; start address
3299 0000D216 29FB        <1>      sub     ebx, edi
3300 0000D218 6643        <1>      inc     bx
3301 0000D21A 66891D[AC4E0100] <1>      mov     [u.r0], bx
3302 0000D221 E9E9F1FFFF <1>      jmp     sysret
3303                                <1>
3304                                <1> sysvideo_26:
3305 0000D226 80FB05        <1>      cmp     bl, 5
3306 0000D229 0F8795000000 <1>      ja      sysvideo_29
3307                                <1>
3308                                <1>      ; BL = 5 = window copy (user to system)
3309                                <1>
3310 0000D22F B800800B00 <1>      mov     eax, 0B8000h
3311 0000D234 39C7        <1>      cmp     edi, eax
3312 0000D236 0F82D3F1FFFF <1>      jb      sysret
3313 0000D23C 6605FF7F <1>      add     ax, 7FFFh ; 32767
3314 0000D240 39C7        <1>      cmp     edi, eax
3315 0000D242 0F87C7F1FFFF <1>      ja      sysret
3316                                <1>
3317                                <1>      ; esi = user buffer (in user's memory space)
3318 0000D248 39CA        <1>      cmp     edx, ecx ; bottom-right > top-left ?
3319 0000D24A 0F865AFEFFFF <1>      jna     sysvideo_15 ; full screen copy
3320                                <1>
3321 0000D250 0FB7C2        <1>      movzx   eax, dx ; bottom right column
3322 0000D253 6629C8        <1>      sub     ax, cx ; - top left column
3323 0000D256 0F82B3F1FFFF <1>      jb      sysret ; invalid
3324 0000D25C 6640        <1>      inc     ax ; same column no == 1 column
3325 0000D25E 50          <1>      push    eax ; byte count per window row
3326                                <1>
3327 0000D25F 52          <1>      push    edx
3328 0000D260 BB40010000 <1>      mov     ebx, 320 ; screen width
3329 0000D265 89C8        <1>      mov     eax, ecx
3330 0000D267 C1E810        <1>      shr     eax, 16 ; top row
3331 0000D26A F7E3        <1>      mul     ebx
3332 0000D26C 6689CA        <1>      mov     dx, cx ; top left column
3333 0000D26F 01D0        <1>      add     eax, edx
3334 0000D271 01C7        <1>      add     edi, eax ; start address
3335 0000D273 59          <1>      pop     ecx ; edx
3336 0000D274 89C8        <1>      mov     eax, ecx
3337 0000D276 C1E810        <1>      shr     eax, 16 ; bottom row
3338 0000D279 F7E3        <1>      mul     ebx
3339 0000D27B 6689CA        <1>      mov     dx, cx ; bottom right column
3340 0000D27E 01D0        <1>      add     eax, edx
3341 0000D280 5A          <1>      pop     edx ; byte count per window row
3342 0000D281 0500800B00 <1>      add     eax, 0B8000h
3343 0000D286 3DFFFF0B00 <1>      cmp     eax, 0BFFFFh
3344 0000D28B 0F877EF1FFFF <1>      ja      sysret
3345 0000D291 57          <1>      push    edi ; start address
3346 0000D292 50          <1>      push    eax ; stop address (included)
3347                                <1> sysvideo_27:
3348 0000D293 89D1        <1>      mov     ecx, edx ; byte count
3349                                <1>      ; user to system video/display page window transfer
3350                                <1>      ; esi = user buffer
3351 0000D295 E880150000 <1>      call    transfer_from_user_buffer ; fast transfer
3352 0000D29A 7221        <1>      jc      short sysvideo_28
3353 0000D29C 010D[AC4E0100] <1>      add     [u.r0], ecx
3354 0000D2A2 01DF        <1>      add     edi, ebx ; next row
3355 0000D2A4 01CE        <1>      add     esi, ecx
3356 0000D2A6 3B3C24        <1>      cmp     edi, [esp] ; stop addr(included in loop)
3357 0000D2A9 76E8        <1>      jna     short sysvideo_27
3358 0000D2AB 5B          <1>      pop     ebx ; stop address
3359 0000D2AC 5F          <1>      pop     edi ; start address
3360 0000D2AD 29FB        <1>      sub     ebx, edi
3361 0000D2AF 6643        <1>      inc     bx
3362 0000D2B1 66891D[AC4E0100] <1>      mov     [u.r0], bx
3363 0000D2B8 E952F1FFFF <1>      jmp     sysret
3364                                <1> sysvideo_28:
3365 0000D2BD 58          <1>      pop     eax
3366 0000D2BE 5A          <1>      pop     edx
3367 0000D2BF E94BF1FFFF <1>      jmp     sysret
3368                                <1>
3369                                <1> sysvideo_29:
3370 0000D2C4 80FB06        <1>      cmp     bl, 6
3371 0000D2C7 0F8797000000 <1>      ja      sysvideo_32
3372                                <1>
3373                                <1>      ; BL = 6 = window copy (system to user)
3374                                <1>
3375 0000D2CD 89F7        <1>      mov     edi, esi ; user buffer
3376                                <1>
3377 0000D2CF B800800B00 <1>      mov     eax, 0B8000h
3378 0000D2D4 39C6        <1>      cmp     esi, eax
3379 0000D2D6 0F8233F1FFFF <1>      jb      sysret
3380 0000D2DC 6605FF7F <1>      add     ax, 7FFFh ; 32767
3381 0000D2E0 39C6        <1>      cmp     esi, eax
3382 0000D2E2 0F8727F1FFFF <1>      ja      sysret
3383                                <1>
3384                                <1>      ; edi = user buffer (in user's memory space)
3385 0000D2E8 39CA        <1>      cmp     edx, ecx ; bottom-right > top-left ?
3386 0000D2EA 0F86E2FDFFFF <1>      jna     sysvideo_17 ; full screen copy
3387                                <1>
3388 0000D2F0 0FB7C2        <1>      movzx   eax, dx ; bottom right column
3389 0000D2F3 6629C8        <1>      sub     ax, cx ; - top left column
3390 0000D2F6 0F8213F1FFFF <1>      jb      sysret ; invalid
3391 0000D2FC 6640        <1>      inc     ax ; same column no == 1 column
3392 0000D2FE 50          <1>      push    eax ; byte count per window row
3393                                <1>
3394 0000D2FF 52          <1>      push    edx
3395 0000D300 BB40010000 <1>      mov     ebx, 320 ; screen width
3396 0000D305 89C8        <1>      mov     eax, ecx
3397 0000D307 C1E810        <1>      shr     eax, 16 ; top row
3398 0000D30A F7E3        <1>      mul     ebx
3399 0000D30C 6689CA        <1>      mov     dx, cx ; top left column

```

```

3400 0000D30F 01D0      <1>      add     eax, edx
3401 0000D311 01C6      <1>      add     esi, eax ; start address
3402 0000D313 59        <1>      pop     ecx ; edx
3403 0000D314 89C8      <1>      mov     eax, ecx
3404 0000D316 C1E810     <1>      shr     eax, 16 ; bottom row
3405 0000D319 F7E3      <1>      mul     ebx
3406 0000D31B 6689CA     <1>      mov     dx, cx ; bottom right column
3407 0000D31E 01D0      <1>      add     eax, edx
3408 0000D320 5A        <1>      pop     edx ; byte count per window row
3409 0000D321 0500800B00 <1>      add     eax, 0B8000h
3410 0000D326 3DFFFF0B00 <1>      cmp     eax, 0BFFFFh
3411 0000D32B 0F87DEF0FFFF <1>      ja      sysret
3412 0000D331 56        <1>      push    esi ; start address
3413 0000D332 50        <1>      push    eax ; stop address (included)
3414                                     <1> sysvideo_30:
3415 0000D333 89D1      <1>      mov     ecx, edx ; byte count
3416                                     <1>      ; user to system video/display page window transfer
3417                                     <1>      ; esi =      user buffer
3418 0000D335 E896140000 <1>      call   transfer_to_user_buffer ; fast transfer
3419 0000D33A 7221      <1>      jc      short sysvideo_31
3420 0000D33C 010D[AC4E0100] <1>      add     [u.r0], ecx
3421 0000D342 01DF      <1>      add     edi, ebx ; next row
3422 0000D344 01CE      <1>      add     esi, ecx
3423 0000D346 3B3C24     <1>      cmp     edi, [esp] ; stop addr(included in loop)
3424 0000D349 76E8      <1>      jna     short sysvideo_30
3425 0000D34B 5B        <1>      pop     ebx ; stop address
3426 0000D34C 5F        <1>      pop     edi ; start address
3427 0000D34D 29FB      <1>      sub     ebx, edi
3428 0000D34F 6643      <1>      inc     bx
3429 0000D351 66891D[AC4E0100] <1>      mov     [u.r0], bx
3430 0000D358 E9B2F0FFFF <1>      jmp     sysret
3431                                     <1> sysvideo_31:
3432 0000D35D 58        <1>      pop     eax
3433 0000D35E 5A        <1>      pop     edx
3434 0000D35F E9ABF0FFFF <1>      jmp     sysret
3435                                     <1>
3436                                     <1> sysvideo_32:
3437 0000D364 80FB07     <1>      cmp     bl, 7
3438 0000D367 770F      <1>      ja      short sysvideo_34
3439                                     <1>
3440                                     <1>      ; BL = 7 = AND display page bytes with CL
3441                                     <1>
3442 0000D369 BE00800B00 <1>      mov     esi, 0B8000h
3443 0000D36E B900800000 <1>      mov     ecx, 32768
3444                                     <1> sysvideo_33:
3445 0000D373 200E      <1>      and     byte [esi], cl
3446 0000D375 46        <1>      inc     esi
3447 0000D376 E2FB      <1>      loop    sysvideo_33
3448                                     <1>
3449                                     <1> sysvideo_34:
3450 0000D378 80FB08     <1>      cmp     bl, 8
3451 0000D37B 770F      <1>      ja      short sysvideo_36
3452                                     <1>
3453                                     <1>      ; BL = 8 = OR display page bytes with CL
3454                                     <1>
3455 0000D37D BE00800B00 <1>      mov     esi, 0B8000h
3456 0000D382 B900800000 <1>      mov     ecx, 32768
3457                                     <1> sysvideo_35:
3458 0000D387 080E      <1>      or      byte [esi], cl
3459 0000D389 46        <1>      inc     esi
3460 0000D38A E2FB      <1>      loop    sysvideo_35
3461                                     <1>
3462                                     <1> sysvideo_36:
3463 0000D38C 80FB09     <1>      cmp     bl, 9
3464 0000D38F 0F877AF0FFFF <1>      ja      sysret ; nothing to do
3465                                     <1>
3466                                     <1>      ; BL = 9 = XOR display page bytes with CL
3467                                     <1>
3468 0000D395 BE00800B00 <1>      mov     esi, 0B8000h
3469 0000D39A B900800000 <1>      mov     ecx, 32768
3470                                     <1> sysvideo_37:
3471 0000D39F 300E      <1>      xor     byte [esi], cl
3472 0000D3A1 46        <1>      inc     esi
3473 0000D3A2 E2FB      <1>      loop    sysvideo_37
3474                                     <1>
3475                                     <1> sysvideo_38:
3476 0000D3A4 80FF02     <1>      cmp     bh, 2
3477 0000D3A7 0F8733030000 <1>      ja      sysvideo_64
3478                                     <1>      ; BH = 2 = VGA Graphics (0A0000h) data transfers
3479                                     <1>
3480 0000D3AD 88DC      <1>      mov     ah, bl
3481 0000D3AF 80E30F     <1>      and     bl, 0Fh
3482 0000D3B2 C0EC04     <1>      shr     ah, 4
3483 0000D3B5 C1E310     <1>      shl     ebx, 16
3484 0000D3B8 66BB4001 <1>      mov     bx, 320 ; 320*200, 320*240
3485 0000D3BC 20E4      <1>      and     ah, ah
3486 0000D3BE 7413      <1>      jz      short sysvideo_39
3487 0000D3C0 66D1E3     <1>      shl     bx, 1 ; 640*200, 640 * 400, 640*480
3488 0000D3C3 80FC02     <1>      cmp     ah, 2
3489 0000D3C6 720B      <1>      jb      short sysvideo_39
3490 0000D3C8 0F8741F0FFFF <1>      ja      sysret ; invalid
3491                                     <1>      ; 800*600
3492 0000D3CE 6681C3A000 <1>      add     bx, 160 ; 800
3493                                     <1> sysvideo_39:
3494 0000D3D3 C1CB10     <1>      ror     ebx, 16
3495                                     <1>
3496 0000D3D6 20DB      <1>      and     bl, bl
3497 0000D3D8 7519      <1>      jnz     short sysvideo_40
3498                                     <1>
3499                                     <1>      ; BL = 0 = Fill color (color in CL] (64K)
3500                                     <1>
3501 0000D3DA 88C8      <1>      mov     al, cl

```

```

3502 0000D3DC B900000100      <1>      mov     ecx, 65536
3503 0000D3E1 890D[AC4E0100]  <1>      mov     [u.r0], ecx
3504 0000D3E7 BF000000A00      <1>      mov     edi, 0A0000h
3505 0000D3EC F3AB          <1>      rep     stosd
3506 0000D3EE E91CF0FFFF      <1>      jmp     sysret
3507                                <1>
3508                                <1> sysvideo_40:
3509 0000D3F3 80FB01      <1>      cmp     bl, 1
3510 0000D3F6 7722          <1>      ja     short sysvideo_42
3511                                <1>
3512 0000D3F8 89CE          <1>      mov     esi, ecx ; user buffer
3513                                <1>      ; BL = 1 = user to system video/display page transfer
3514                                <1> sysvideo_41:
3515 0000D3FA BF000000A00      <1>      mov     edi, 0A0000h
3516                                <1>      ; edi = video page address
3517 0000D3FF B900000100      <1>      mov     ecx, 65536
3518 0000D404 E811140000      <1>      call    transfer_from_user_buffer ; fast transfer
3519 0000D409 0F8200F0FFFF      <1>      jc     sysret ; [u.r0] = 0
3520 0000D40F 890D[AC4E0100]  <1>      mov     [u.r0], ecx
3521 0000D415 E9F5EFFFFF      <1>      jmp     sysret
3522                                <1>
3523                                <1> sysvideo_42:
3524 0000D41A 80FB02      <1>      cmp     bl, 2
3525 0000D41D 7722          <1>      ja     short sysvideo_44
3526                                <1>
3527 0000D41F 89CF          <1>      mov     edi, ecx ; user buffer
3528                                <1>      ; BL = 2 = system to user video/display page transfer
3529                                <1> sysvideo_43:
3530 0000D421 BE000000A00      <1>      mov     esi, 0A0000h
3531 0000D426 B900000100      <1>      mov     ecx, 65536
3532 0000D42B E8A0130000      <1>      call    transfer_to_user_buffer ; fast transfer
3533 0000D430 0F82D9EFFFFF      <1>      jc     sysret ; [u.r0] = 0
3534 0000D436 890D[AC4E0100]  <1>      mov     [u.r0], ecx
3535 0000D43C E9CEEFFFFF      <1>      jmp     sysret
3536                                <1>
3537                                <1> sysvideo_44:
3538 0000D441 80FB03      <1>      cmp     bl, 3
3539 0000D444 777A          <1>      ja     short sysvideo_49
3540                                <1>
3541                                <1>      ; BL = 3 = NOT bits in window (ECX, EDX)
3542                                <1>
3543 0000D446 BF000000A00      <1>      mov     edi, 0A0000h
3544 0000D44B 89FE          <1>      mov     esi, edi
3545                                <1>
3546 0000D44D 39CA          <1>      cmp     edx, ecx ; bottom-right > top-left ?
3547 0000D44F 770B          <1>      ja     short sysvideo_45 ; window
3548                                <1>      ; full screen (update)
3549 0000D451 B900000100      <1>      mov     ecx, 65536
3550 0000D456 890D[AC4E0100]  <1>      mov     [u.r0], ecx
3551                                <1> sysvideo_45:
3552 0000D45C F616          <1>      not     byte [esi] ; NOT operation
3553 0000D45E 46            <1>      inc     esi
3554 0000D45F E2FB          <1>      loop    sysvideo_45
3555 0000D461 E9A9EFFFFF      <1>      jmp     sysret
3556                                <1> sysvideo_46:
3557 0000D466 0FB7C2      <1>      movzx   eax, dx ; bottom right column
3558 0000D469 6629C8      <1>      sub     ax, cx ; - top left column
3559 0000D46C 0F829DEFFFFF      <1>      jb     sysret ; invalid
3560 0000D472 6640          <1>      inc     ax ; same column no == 1 column
3561 0000D474 50            <1>      push    eax ; byte count per window row
3562 0000D475 52            <1>      push    edx
3563 0000D476 C1EB10      <1>      shr     ebx, 16 ; 320,640,800 : screen width
3564 0000D479 89C8          <1>      mov     eax, ecx
3565 0000D47B C1E810      <1>      shr     eax, 16 ; top row
3566 0000D47E F7E3          <1>      mul     ebx
3567 0000D480 6689CA      <1>      mov     dx, cx ; top left column
3568 0000D483 01D0          <1>      add     eax, edx
3569 0000D485 01C6          <1>      add     esi, eax ; start address
3570 0000D487 59            <1>      pop     ecx ; edx
3571 0000D488 89C8          <1>      mov     eax, ecx
3572 0000D48A C1E810      <1>      shr     eax, 16 ; bottom row
3573 0000D48D F7E3          <1>      mul     ebx
3574 0000D48F 6689CA      <1>      mov     dx, cx ; bottom right column
3575 0000D492 01D0          <1>      add     eax, edx
3576 0000D494 01C7          <1>      add     edi, eax ; stop address (included)
3577 0000D496 5A            <1>      pop     edx ; byte count per window row
3578 0000D497 81FFFFFF0A00      <1>      cmp     edi, 0AFFFFFh
3579 0000D49D 0F876CEFFFFF      <1>      ja     sysret
3580 0000D4A3 56            <1>      push    esi
3581 0000D4A4 4E            <1>      dec     esi
3582                                <1> sysvideo_47:
3583 0000D4A5 89D1          <1>      mov     ecx, edx
3584                                <1> sysvideo_48:
3585 0000D4A7 46            <1>      inc     esi
3586 0000D4A8 F616          <1>      not     byte [esi]
3587 0000D4AA E2FB          <1>      loop    sysvideo_48
3588 0000D4AC 01DE          <1>      add     esi, ebx ; bytes per screen row
3589                                <1>      ;
3590 0000D4AE 39FE          <1>      cmp     esi, edi ; stop address (included in loop)
3591 0000D4B0 76F3          <1>      jna     short sysvideo_47
3592 0000D4B2 5E            <1>      pop     esi
3593 0000D4B3 29F7          <1>      sub     edi, esi
3594 0000D4B5 893D[AC4E0100]  <1>      mov     [u.r0], edi
3595 0000D4BB E94FEFFFFF      <1>      jmp     sysret
3596                                <1>
3597                                <1> sysvideo_49:
3598 0000D4C0 80FB04      <1>      cmp     bl, 4
3599 0000D4C3 0F87A1000000      <1>      ja     sysvideo_52
3600                                <1>
3601                                <1>      ; BL = 4 = window copy (system to system)
3602                                <1>
3603 0000D4C9 B8000000A00      <1>      mov     eax, 0A0000h

```

```

3604 0000D4CE 39C6      <1>      cmp     esi, eax
3605 0000D4D0 0F8239FFFFFF      <1>      jb      sysret
3606 0000D4D6 39C7      <1>      cmp     edi, eax
3607 0000D4D8 0F8231FFFFFF      <1>      jb      sysret
3608 0000D4DE 6683C0FF      <1>      add     ax, 0FFFFh ; 65535
3609 0000D4E2 39C6      <1>      cmp     esi, eax
3610 0000D4E4 0F8725FFFFFF      <1>      ja      sysret
3611 0000D4EA 39C7      <1>      cmp     edi, eax
3612 0000D4EC 0F871DEFFFFFF      <1>      ja      sysret
3613                                     <1>
3614 0000D4F2 39CA      <1>      cmp     edx, ecx ; bottom-right > top-left ?
3615 0000D4F4 7712      <1>      ja      short sysvideo_50 ; window
3616                                     <1>      ; full screen copy
3617 0000D4F6 89C1      <1>      mov     ecx, eax
3618 0000D4F8 29F9      <1>      sub     ecx, edi
3619 0000D4FA 41         <1>      inc     ecx
3620 0000D4FB 890D[AC4E0100]    <1>      mov     [u.r0], ecx
3621 0000D501 F3A4      <1>      rep     movsb
3622 0000D503 E907EFFFFFF      <1>      jmp     sysret
3623                                     <1> sysvideo_50:
3624 0000D508 0FB7C2      <1>      movzx   eax, dx ; bottom right column
3625 0000D50B 6629C8      <1>      sub     ax, cx ; - top left column
3626 0000D50E 0F82FBEEFFFF      <1>      jb      sysret ; invalid
3627 0000D514 6640      <1>      inc     ax ; same column no == 1 column
3628 0000D516 50         <1>      push    eax ; byte count per window row
3629                                     <1>      ;
3630 0000D517 52         <1>      push    edx
3631 0000D518 C1EB10      <1>      shr     ebx, 16 ; 320,640,800 : screen width
3632 0000D51B 89C8      <1>      mov     eax, ecx
3633 0000D51D C1E810      <1>      shr     eax, 16 ; top row
3634 0000D520 F7E3      <1>      mul     ebx
3635 0000D522 6689CA      <1>      mov     dx, cx ; top left column
3636 0000D525 01D0      <1>      add     eax, edx
3637 0000D527 01C7      <1>      add     edi, eax ; start address
3638 0000D529 01C6      <1>      add     esi, eax
3639 0000D52B 59         <1>      pop     ecx ; edx
3640 0000D52C 89C8      <1>      mov     eax, ecx
3641 0000D52E C1E810      <1>      shr     eax, 16 ; bottom row
3642 0000D531 F7E3      <1>      mul     ebx
3643 0000D533 6689CA      <1>      mov     dx, cx ; bottom right column
3644 0000D536 01D0      <1>      add     eax, edx
3645 0000D538 5A         <1>      pop     edx ; byte count per window row
3646 0000D539 0500000A00    <1>      add     eax, 0A0000h
3647 0000D53E 3DFFFF0A00    <1>      cmp     eax, 0AFFFFh
3648 0000D543 0F87C6EEFFFF      <1>      ja      sysret
3649 0000D549 57         <1>      push    edi ; start address
3650 0000D54A 50         <1>      push    eax ; stop address (included)
3651                                     <1> sysvideo_51:
3652 0000D54B 89D1      <1>      mov     ecx, edx
3653 0000D54D F3A4      <1>      rep     movsb
3654 0000D54F 4F         <1>      dec     edi
3655 0000D550 4E         <1>      dec     esi
3656 0000D551 01DF      <1>      add     edi, ebx ; bytes per screen row
3657 0000D553 01DE      <1>      add     esi, ebx
3658                                     <1>      ;
3659 0000D555 3B3C24      <1>      cmp     edi, [esp] ; stop addr(included in loop)
3660 0000D558 76F1      <1>      jna     short sysvideo_51
3661 0000D55A 5B         <1>      pop     ebx ; stop address
3662 0000D55B 5F         <1>      pop     edi ; start address
3663 0000D55C 29FB      <1>      sub     ebx, edi
3664 0000D55E 43         <1>      inc     ebx
3665 0000D55F 891D[AC4E0100]    <1>      mov     [u.r0], ebx
3666 0000D565 E9A5EEFFFF      <1>      jmp     sysret
3667                                     <1>
3668                                     <1> sysvideo_52:
3669 0000D56A 80FB05      <1>      cmp     bl, 5
3670 0000D56D 0F8791000000    <1>      ja      sysvideo_55
3671                                     <1>
3672                                     <1>      ; BL = 5 = window copy (user to system)
3673                                     <1>
3674 0000D573 B800000A00    <1>      mov     eax, 0A0000h
3675 0000D578 39C7      <1>      cmp     edi, eax
3676 0000D57A 0F828FEEFFFF      <1>      jb      sysret
3677 0000D580 6683C0FF      <1>      add     ax, 0FFFFh ; 65535
3678 0000D584 39C7      <1>      cmp     edi, eax
3679 0000D586 0F8783EEFFFF      <1>      ja      sysret
3680                                     <1>
3681                                     <1>      ; esi = user buffer (in user's memory space)
3682 0000D58C 39CA      <1>      cmp     edx, ecx ; bottom-right > top-left ?
3683 0000D58E 0F8666FEFFFF      <1>      jna     sysvideo_41 ; full screen copy
3684                                     <1>
3685 0000D594 0FB7C2      <1>      movzx   eax, dx ; bottom right column
3686 0000D597 6629C8      <1>      sub     ax, cx ; - top left column
3687 0000D59A 0F826FEEFFFF      <1>      jb      sysret ; invalid
3688 0000D5A0 6640      <1>      inc     ax ; same column no == 1 column
3689 0000D5A2 50         <1>      push    eax ; byte count per window row
3690                                     <1>
3691 0000D5A3 52         <1>      push    edx
3692 0000D5A4 C1EB10      <1>      shr     ebx, 16 ; 320,640,800 : screen width
3693 0000D5A7 89C8      <1>      mov     eax, ecx
3694 0000D5A9 C1E810      <1>      shr     eax, 16 ; top row
3695 0000D5AC F7E3      <1>      mul     ebx
3696 0000D5AE 6689CA      <1>      mov     dx, cx ; top left column
3697 0000D5B1 01D0      <1>      add     eax, edx
3698 0000D5B3 01C7      <1>      add     edi, eax ; start address
3699 0000D5B5 59         <1>      pop     ecx ; edx
3700 0000D5B6 89C8      <1>      mov     eax, ecx
3701 0000D5B8 C1E810      <1>      shr     eax, 16 ; bottom row
3702 0000D5BB F7E3      <1>      mul     ebx
3703 0000D5BD 6689CA      <1>      mov     dx, cx ; bottom right column
3704 0000D5C0 01D0      <1>      add     eax, edx
3705 0000D5C2 5A         <1>      pop     edx ; byte count per window row

```



```

3706 0000D5C3 0500000A00      <1>      add     eax, 0A0000h
3707 0000D5C8 3DFFFF0A00      <1>      cmp     eax, 0AFFFFh
3708 0000D5CD 0F873CEEFFFF      <1>      ja      sysret
3709 0000D5D3 57      <1>      push    edi ; start address
3710 0000D5D4 50      <1>      push    eax ; stop address (included)
3711      <1> sysvideo_53:
3712 0000D5D5 89D1      <1>      mov     ecx, edx ; byte count
3713      <1>      ; user to system video/display page window transfer
3714      <1>      ; esi =      user buffer
3715 0000D5D7 E83E120000      <1>      call   transfer_from_user_buffer ; fast transfer
3716 0000D5DC 721F      <1>      jc      short sysvideo_54
3717 0000D5DE 010D[AC4E0100]      <1>      add     [u.r0], ecx
3718 0000D5E4 01DF      <1>      add     edi, ebx ; next row
3719 0000D5E6 01CE      <1>      add     esi, ecx
3720 0000D5E8 3B3C24      <1>      cmp     edi, [esp] ; stop addr(included in loop)
3721 0000D5EB 76E8      <1>      jna     short sysvideo_53
3722 0000D5ED 5B      <1>      pop     ebx ; stop address
3723 0000D5EE 5F      <1>      pop     edi ; start address
3724 0000D5EF 29FB      <1>      sub     ebx, edi
3725 0000D5F1 43      <1>      inc     ebx
3726 0000D5F2 891D[AC4E0100]      <1>      mov     [u.r0], ebx
3727 0000D5F8 E912EEFFFF      <1>      jmp     sysret
3728      <1> sysvideo_54:
3729 0000D5FD 58      <1>      pop     eax
3730 0000D5FE 5A      <1>      pop     edx
3731 0000D5FF E90BEEFFFF      <1>      jmp     sysret
3732      <1>
3733      <1> sysvideo_55:
3734 0000D604 80FB06      <1>      cmp     bl, 6
3735 0000D607 0F8793000000      <1>      ja      sysvideo_58
3736      <1>
3737      <1>      ; BL = 6 = window copy (system to user)
3738      <1>
3739 0000D60D 89F7      <1>      mov     edi, esi ; user buffer
3740      <1>
3741 0000D60F B800000A00      <1>      mov     eax, 0A0000h
3742 0000D614 39C6      <1>      cmp     esi, eax
3743 0000D616 0F82F3EDFFFF      <1>      jb      sysret
3744 0000D61C 6683C0FF      <1>      add     ax, 0FFFFh ; 65535
3745 0000D620 39C6      <1>      cmp     esi, eax
3746 0000D622 0F87E7EDFFFF      <1>      ja      sysret
3747      <1>
3748      <1>      ; edi = user buffer (in user's memory space)
3749 0000D628 39CA      <1>      cmp     edx, ecx ; bottom-right > top-left ?
3750 0000D62A 0F86A2FAFFFF      <1>      jna     sysvideo_17 ; full screen copy
3751      <1>
3752 0000D630 0FB7C2      <1>      movzx   eax, dx ; bottom right column
3753 0000D633 6629C8      <1>      sub     ax, cx ; - top left column
3754 0000D636 0F82D3EDFFFF      <1>      jb      sysret ; invalid
3755 0000D63C 6640      <1>      inc     ax ; same column no == 1 column
3756 0000D63E 50      <1>      push    eax ; byte count per window row
3757      <1>
3758 0000D63F 52      <1>      push    edx
3759 0000D640 C1EB10      <1>      shr     ebx, 16 ; 320, 640,800 ; screen width
3760 0000D643 89C8      <1>      mov     eax, ecx
3761 0000D645 C1E810      <1>      shr     eax, 16 ; top row
3762 0000D648 F7E3      <1>      mul     ebx
3763 0000D64A 6689CA      <1>      mov     dx, cx ; top left column
3764 0000D64D 01D0      <1>      add     eax, edx
3765 0000D64F 01C6      <1>      add     esi, eax ; start address
3766 0000D651 59      <1>      pop     ecx ; edx
3767 0000D652 89C8      <1>      mov     eax, ecx
3768 0000D654 C1E810      <1>      shr     eax, 16 ; bottom row
3769 0000D657 F7E3      <1>      mul     ebx
3770 0000D659 6689CA      <1>      mov     dx, cx ; bottom right column
3771 0000D65C 01D0      <1>      add     eax, edx
3772 0000D65E 5A      <1>      pop     edx ; byte count per window row
3773 0000D65F 0500000A00      <1>      add     eax, 0A0000h
3774 0000D664 3DFFFF0A00      <1>      cmp     eax, 0AFFFFh
3775 0000D669 0F87A0EDFFFF      <1>      ja      sysret
3776 0000D66F 56      <1>      push    esi ; start address
3777 0000D670 50      <1>      push    eax ; stop address (included)
3778      <1> sysvideo_56:
3779 0000D671 89D1      <1>      mov     ecx, edx ; byte count
3780      <1>      ; user to system video/display page window transfer
3781      <1>      ; esi =      user buffer
3782 0000D673 E858110000      <1>      call   transfer_to_user_buffer ; fast transfer
3783 0000D678 721F      <1>      jc      short sysvideo_57
3784 0000D67A 010D[AC4E0100]      <1>      add     [u.r0], ecx
3785 0000D680 01DF      <1>      add     edi, ebx ; next row
3786 0000D682 01CE      <1>      add     esi, ecx
3787 0000D684 3B3C24      <1>      cmp     edi, [esp] ; stop addr(included in loop)
3788 0000D687 76E8      <1>      jna     short sysvideo_56
3789 0000D689 5B      <1>      pop     ebx ; stop address
3790 0000D68A 5F      <1>      pop     edi ; start address
3791 0000D68B 29FB      <1>      sub     ebx, edi
3792 0000D68D 43      <1>      inc     ebx
3793 0000D68E 891D[AC4E0100]      <1>      mov     [u.r0], ebx
3794 0000D694 E976EDFFFF      <1>      jmp     sysret
3795      <1> sysvideo_57:
3796 0000D699 58      <1>      pop     eax
3797 0000D69A 5A      <1>      pop     edx
3798 0000D69B E96FEDFFFF      <1>      jmp     sysret
3799      <1>
3800      <1> sysvideo_58:
3801 0000D6A0 80FB07      <1>      cmp     bl, 7
3802 0000D6A3 770F      <1>      ja      short sysvideo_60
3803      <1>
3804      <1>      ; BL = 7 = AND display page bytes with CL
3805      <1>
3806 0000D6A5 BE00000A00      <1>      mov     esi, 0A0000h
3807 0000D6AA B900000100      <1>      mov     ecx, 65536

```

```

3808
3809 0000D6AF 200E
3810 0000D6B1 46
3811 0000D6B2 E2FB
3812
3813
3814 0000D6B4 80FB08
3815 0000D6B7 770F
3816
3817
3818
3819 0000D6B9 BE00000A00
3820 0000D6BE B900000100
3821
3822 0000D6C3 080E
3823 0000D6C5 46
3824 0000D6C6 E2FB
3825
3826
3827 0000D6C8 80FB09
3828 0000D6CB 0F873EEDFFFF
3829
3830
3831
3832 0000D6D1 BE00000A00
3833 0000D6D6 B900000100
3834
3835 0000D6DB 300E
3836 0000D6DD 46
3837 0000D6DE E2FB
3838
3839
3840 0000D6E0 80FF03
3841 0000D6E3 7460
3842 0000D6E5 80FF04
3843 0000D6E8 771F
3844
3845
3846
3847
3848
3849
3850
3851
3852 0000D6EA B800800B00
3853 0000D6EF B908000000
3854 0000D6F4 E89B7FFFFF
3855 0000D6F9 0F8210EDFFFF
3856
3857 0000D6FF A3[AC4E0100]
3858 0000D704 E906EDFFFF
3859
3860
3861 0000D709 80FF05
3862 0000D70C 771F
3863
3864
3865
3866
3867
3868
3869
3870
3871 0000D70E B800000A00
3872 0000D713 B910000000
3873 0000D718 E8777FFFFF
3874 0000D71D 0F82ECECFFFF
3875
3876 0000D723 A3[AC4E0100]
3877 0000D728 E9E2ECFFFF
3878
3879
3880 0000D72D 80FF06
3881 0000D730 7705
3882
3883
3884
3885
3886
3887
3888 0000D732 E9D8ECFFFF
3889
3890
3891 0000D737 80FF07
3892 0000D73A 0F87CFECFFFF
3893
3894
3895
3896
3897
3898
3899 0000D740 E9CAECFFFF
3900
3901
3902
3903
3904
3905 0000D745 E9C5ECFFFF
3906
3907
3908
3909

<1> sysvideo_59:
<1>     and     byte [esi], cl
<1>     inc     esi
<1>     loop    sysvideo_59
<1>
<1> sysvideo_60:
<1>     cmp     bl, 8
<1>     ja      short sysvideo_62
<1>
<1>         ; BL = 8 = OR display page bytes with CL
<1>
<1>     mov     esi, 0A0000h
<1>     mov     ecx, 65536
<1> sysvideo_61:
<1>     or      byte [esi], cl
<1>     inc     esi
<1>     loop    sysvideo_61
<1>
<1> sysvideo_62:
<1>     cmp     bl, 9
<1>     ja      sysret ; nothing to do
<1>
<1>         ; BL = 9 = XOR display page bytes with CL
<1>
<1>     mov     esi, 0A0000h
<1>     mov     ecx, 65536
<1> sysvideo_63:
<1>     xor     byte [esi], cl
<1>     inc     esi
<1>     loop    sysvideo_63
<1>
<1> sysvideo_64:
<1>     cmp     bh, 3
<1>     je      short sysvideo_68
<1>     cmp     bh, 4
<1>     ja      short sysvideo_65
<1>
<1>         ; BH = 4
<1>         ; Direct User Access for CGA video memory.
<1>         ; Setup user's page tables for direct access to 0B8000h.
<1>         ;
<1>         ; Permission checks are not implemented yet !
<1>         ; (11/07/2016)
<1>
<1>     mov     eax, 0B8000h
<1>     mov     ecx, 8 ; 8 pages (8*4K=32K)
<1>     call    direct_memory_access
<1>     jc      sysret
<1>         ; eax = 0B8000h if there is not an error
<1>     mov     [u.r0], eax
<1>     jmp     sysret
<1>
<1> sysvideo_65:
<1>     cmp     bh, 5
<1>     ja      short sysvideo_66
<1>
<1>         ; BH = 5
<1>         ; Direct User Access for VGA video memory.
<1>         ; Setup user's page tables for direct access to 0A0000h.
<1>         ;
<1>         ; Permission checks are not implemented yet !
<1>         ; (11/07/2016)
<1>
<1>     mov     eax, 0A0000h
<1>     mov     ecx, 16 ; 16 pages (16*4K=64K)
<1>     call    direct_memory_access
<1>     jc      sysret
<1>         ; eax = 0A0000h if there is not an error
<1>     mov     [u.r0], eax
<1>     jmp     sysret
<1>
<1> sysvideo_66:
<1>     cmp     bh, 6
<1>     ja      short sysvideo_67
<1>
<1>         ; BH = 6
<1>         ; Direct User Access for (Super VGA) Linear Frame Buffer.
<1>         ; Setup user's page tables for direct access to LFB.
<1>         ;
<1>         ; Not implemented yet !
<1>         ; (11/07/2016)
<1>     jmp     sysret
<1>
<1> sysvideo_67:
<1>     cmp     bh, 7
<1>     ja      sysret ; invalid !
<1>
<1>         ; BH = 7
<1>         ; Get (Super/Extended VGA) Linear Frame Buffer info.
<1>         ;
<1>         ; Not implemented yet !
<1>         ; (11/07/2016)
<1>     jmp     sysret
<1>
<1> sysvideo_68:
<1>         ; BH = 3
<1>         ; Super VGA, LINEAR FRAME BUFFER data transfers
<1>         ; Not implemented for yet ! (11/07/2016)
<1>     jmp     sysret
<1>
<1> sysaudio: ; AUDIO FUNCTIONS
<1>         ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
<1>

```

```

3910      <1>      ; Inputs:
3911      <1>      ;      BH = 0 -> Beep (PC Speaker)
3912      <1>      ;      BL = Duration Counter (1 for 1/64 second)
3913      <1>      ;      CX = Frequency Divisor (1193180/Frequency)
3914      <1>      ;      (1331 for 886 Hz)
3915      <1>      ;
3916      <1>      ; Outputs:
3917      <1>      ;      none
3918      <1>      ;
3919      <1>      and    bh, bh
3920      <1>      jnz    sysret
3921      <1>      call   beep
3922      <1>      jmp     sysret
3923      <1>
3924      <1> syslink:
3925      <1>      ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
3926      <1>      ; temporary !
3927      <1>      mov    eax, ERR_INV_FNUMBER ; 'invalid function number !'
3928      <1>      mov    [u.error], eax
3929      <1>      mov    [u.r0], eax
3930      <1>      jmp     error
3931      <1>
3932      <1> isdir:
3933      <1>      ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
3934      <1>      ; 04/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
3935      <1>      ;
3936      <1>      ; 'isdir' check to see if the i-node whose i-number is in r1
3937      <1>      ; is a directory. If it is, an error occurs, because 'isdir'
3938      <1>      ; called by syslink and sysunlink to make sure directories
3939      <1>      ; are not linked. If the user is the super user (u.uid=0),
3940      <1>      ; 'isdir' does not bother checking. The current i-node
3941      <1>      ; is not disturbed.
3942      <1>      ;
3943      <1>      ; INPUTS ->
3944      <1>      ;      r1 - contains the i-number whose i-node is being checked.
3945      <1>      ;      u.uid - user id
3946      <1>      ; OUTPUTS ->
3947      <1>      ;      r1 - contains current i-number upon exit
3948      <1>      ;      (current i-node back in core)
3949      <1>      ;
3950      <1>      ; ((AX = R1))
3951      <1>      ;
3952      <1>      ; ((Modified registers: EAX, EDX, EBX, ECX, ESI, EDI, EBP))
3953      <1>      ;
3954      <1>
3955      <1>      ; / if the i-node whose i-number is in r1 is a directory
3956      <1>      ; / there is an error unless super user made the call
3957      <1>
3958      <1>      cmp     byte [u.uid], 0
3959      <1>      ; tstb u.uid / super user
3960      <1>      jna     short isdir1
3961      <1>      ; beq lf / yes, don't care
3962      <1>      push    word [ii]
3963      <1>      ; mov ii, -(sp) / put current i-number on stack
3964      <1>      call    iget
3965      <1>      ; jsr r0, iget / get i-node into core (i-number in r1)
3966      <1>      test    word [i.flgs], 4000h ; Bit 14 : Directory flag
3967      <1>      ; bit $40000, i.flgs / is it a directory
3968      <1>      ; jnz     error
3969      <1>      ; bne error9 / yes, error
3970      <1>      jz      short isdir0
3971      <1>      mov     dword [u.error], ERR_NOT_FILE ; 11 ; ERR_DIR_ACCESS
3972      <1>      ; 'permission denied !' error
3973      <1>      ; pop     ax
3974      <1>      jmp     error
3975      <1> isdir0:
3976      <1>      pop     ax
3977      <1>      ; mov (sp)+, r1 / no, put current i-number in r1 (ii)
3978      <1>      call    iget
3979      <1>      ; jsr r0, iget / get it back in
3980      <1> isdir1: ; 1:
3981      <1>      retn
3982      <1>      ; rts r0
3983      <1>
3984      <1> sysunlink:
3985      <1>      ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
3986      <1>      ; temporary !
3987      <1>      mov     eax, ERR_INV_FNUMBER ; 'invalid function number !'
3988      <1>      mov     [u.error], eax
3989      <1>      mov     [u.r0], eax
3990      <1>      jmp     error
3991      <1> mkdir:
3992      <1>      ; 04/12/2015 (14 byte directory names)
3993      <1>      ; 12/10/2015
3994      <1>      ; 17/06/2015 (Retro UNIX 386 v1 - Beginning)
3995      <1>      ; 29/04/2013 - 01/08/2013 (Retro UNIX 8086 v1)
3996      <1>      ;
3997      <1>      ; 'mkdir' makes a directory entry from the name pointed to
3998      <1>      ; by u.namep into the current directory.
3999      <1>      ;
4000      <1>      ; INPUTS ->
4001      <1>      ;      u.namep - points to a file name
4002      <1>      ;      that is about to be a directory entry.
4003      <1>      ;      ii - current directory's i-number.
4004      <1>      ; OUTPUTS ->
4005      <1>      ;      u.dirbuf+2 - u.dirbuf+10 - contains file name.
4006      <1>      ;      u.off - points to entry to be filled
4007      <1>      ;      in the current directory
4008      <1>      ;      u.base - points to start of u.dirbuf.
4009      <1>      ;      r1 - contains i-number of current directory

```

```

4010      <1>      ;
4011      <1>      ; ((AX = R1)) output
4012      <1>      ;
4013      <1>      ; (Retro UNIX Prototype : 11/11/2012, UNIXCOPY.ASM)
4014      <1>      ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
4015      <1>      ;
4016      <1>
4017      <1>      ; 17/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
4018      <1>      xor     eax, eax
4019      <1>      mov     edi, u.dirbuf+2
4020      <1>      mov     esi, edi
4021      <1>      stosd
4022      <1>      stosd
4023      <1>      ; 04/12/2015 (14 byte directory names)
4024      <1>      stosd
4025      <1>      stosw
4026      <1>      ; jsr r0,copyz; u.dirbuf+2; u.dirbuf+10. / clear this
4027      <1>      mov     edi, esi ; offset to u.dirbuf
4028      <1>      ; 12/10/2015 ([u.namep] -> ebp)
4029      <1>      ;mov     ebp, [u.namep]
4030      <1>      call    trans_addr_nmbp ; convert virtual address to physical
4031      <1>      ; esi = physical address (page start + offset)
4032      <1>      ; ecx = byte count in the page (1 - 4096)
4033      <1>      ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
4034      <1>      ; mov u.namep,r2 / r2 points to name of directory entry
4035      <1>      ; mov $u.dirbuf+2,r3 / r3 points to u.dirbuf+2
4036      <1>      mkdir_1: ; 1:
4037      <1>      inc     ebp ; 12/10/2015
4038      <1>      ;
4039      <1>      ; / put characters in the directory name in u.dirbuf+2 - u.dirbuf+10
4040      <1>      ; 01/08/2013
4041      <1>      lodsb
4042      <1>      ; movb (r2)+,r1 / move character in name to r1
4043      <1>      and     al, al
4044      <1>      jz      short mkdir_3
4045      <1>      ; beq lf / if null, done
4046      <1>      cmp     al, '/'
4047      <1>      ; cmp r1,$' / is it a "/"?
4048      <1>      je      short mkdir_err
4049      <1>      ;je      error
4050      <1>      ; beq error9 / yes, error
4051      <1>      ; 12/10/2015
4052      <1>      dec     cx
4053      <1>      jnz     short mkdir_2
4054      <1>      ; 12/10/2015 ([u.namep] -> ebp)
4055      <1>      call    trans_addr_nm ; convert virtual address to physical
4056      <1>      ; esi = physical address (page start + offset)
4057      <1>      ; ecx = byte count in the page
4058      <1>      ; edi = offset to u.dirbuf (edi is not modified in trans_addr_nm)
4059      <1>      mkdir_2:
4060      <1>      cmp     edi, u.dirbuf+16 ; ; 04/12/2015 (10 -> 16)
4061      <1>      ; cmp r3,$u.dirbuf+10. / have we reached the last slot for
4062      <1>      ; / a char?
4063      <1>      je      short mkdir_1
4064      <1>      ; beq lb / yes, go back
4065      <1>      stosb
4066      <1>      ; movb r1,(r3)+ / no, put the char in the u.dirbuf
4067      <1>      jmp     short mkdir_1
4068      <1>      ; br lb / get next char
4069      <1>      mkdir_err:
4070      <1>      ; 17/06/2015
4071      <1>      mov     dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
4071      <1>      0000D7F6 0000
4072      <1>      jmp     error
4073      <1>
4074      <1>      mkdir_3: ; 1:
4075      <1>      mov     eax, [u.dirp]
4076      <1>      mov     [u.off], eax
4077      <1>      ; mov u.dirp,u.off / pointer to empty current directory
4078      <1>      ; / slot to u.off
4079      <1>      wdir: ; 29/04/2013
4080      <1>      mov     dword [u.base], u.dirbuf
4080      <1>      0000D80D [E04E0100]
4081      <1>      ; mov $u.dirbuf,u.base / u.base points to created file name
4082      <1>      mov     dword [u.count], 16 ; 04/12/2015 (10 -> 16)
4082      <1>      0000D819 0000
4083      <1>      ; mov $10.,u.count / u.count = 10
4084      <1>      mov     ax, [ii]
4085      <1>      ; mov ii,r1 / r1 has i-number of current directory
4086      <1>      mov     dl, 1 ; owner flag mask ; RETRO UNIX 8086 v1 modification !
4087      <1>      call    access
4088      <1>      ; jsr r0,access; 1 / get i-node and set its file up
4089      <1>      ; / for writing
4090      <1>      ; AX = i-number of current directory
4091      <1>      ; 01/08/2013
4092      <1>      inc     byte [u.kcall] ; the caller is 'mkdir' sign
4093      <1>      call    writei
4094      <1>      ; jsr r0,writei / write into directory
4095      <1>      retn
4096      <1>      ; rts r0
4097      <1>
4098      <1>      sysexec:
4099      <1>      ; 04/01/2017
4100      <1>      ; 24/10/2016
4101      <1>      ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)
4102      <1>      ; 23/06/2015 - 23/10/2015 (Retro UNIX 386 v1)
4103      <1>      ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
4104      <1>      ;
4105      <1>      ; 'sysexec' initiates execution of a file whose path name if
4106      <1>      ; pointed to by 'name' in the sysexec call.
4107      <1>      ; 'sysexec' performs the following operations:
4108      <1>      ; 1. obtains i-number of file to be executed via 'namei'.

```



```

4109      <1>      ;      2. obtains i-node of file to be exceuted via 'iget'.
4110      <1>      ;      3. sets trap vectors to system routines.
4111      <1>      ;      4. loads arguments to be passed to executing file into
4112      <1>      ;      highest locations of user's core
4113      <1>      ;      5. puts pointers to arguments in locations immediately
4114      <1>      ;      following arguments.
4115      <1>      ;      6.      saves number of arguments in next location.
4116      <1>      ;      7. intializes user's stack area so that all registers
4117      <1>      ;      will be zeroed and the PS is cleared and the PC set
4118      <1>      ;      to core when 'sysret' restores registers
4119      <1>      ;      and does an rti.
4120      <1>      ;      8. inializes u.r0 and u.sp
4121      <1>      ;      9. zeros user's core down to u.r0
4122      <1>      ;      10.     reads executable file from storage device into core
4123      <1>      ;      starting at location 'core'.
4124      <1>      ;      11.     sets u.break to point to end of user's code with
4125      <1>      ;      data area appended.
4126      <1>      ;      12.     calls 'sysret' which returns control at location
4127      <1>      ;      'core' via 'rti' instruction.
4128      <1>      ;
4129      <1>      ; Calling sequence:
4130      <1>      ;      sysexec; namep; argp
4131      <1>      ; Arguments:
4132      <1>      ;      namep - points to pathname of file to be executed
4133      <1>      ;      argp  - address of table of argument pointers
4134      <1>      ;      argp1... argpn - table of argument pointers
4135      <1>      ;      argp1:<...0> ... argpn:<...0> - argument strings
4136      <1>      ; Inputs: (arguments)
4137      <1>      ; Outputs: -
4138      <1>      ; .....
4139      <1>      ;
4140      <1>      ; Retro UNIX 386 v1 modification:
4141      <1>      ;      User application runs in it's own virtual space
4142      <1>      ;      which is izolated from kernel memory (and other
4143      <1>      ;      memory pages) via 80386  paging in ring 3
4144      <1>      ;      privilige mode. Virtual start address is always 0.
4145      <1>      ;      User's core memory starts at linear address 400000h
4146      <1>      ;      (the end of the 1st 4MB).
4147      <1>      ;
4148      <1>      ; Retro UNIX 8086 v1 modification:
4149      <1>      ;      user/application segment and system/kernel segment
4150      <1>      ;      are different and sysenter/sysret/sysrele routines
4151      <1>      ;      are different (user's registers are saved to
4152      <1>      ;      and then restored from system's stack.)
4153      <1>      ;
4154      <1>      ;      NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
4155      <1>      ;      arguments which were in these registers;
4156      <1>      ;      but, it returns by putting the 1st argument
4157      <1>      ;      in 'u.namep' and the 2nd argument
4158      <1>      ;      on top of stack. (1st argument is offset of the
4159      <1>      ;      file/path name in the user's program segment.)
4160      <1>      ;
4161      <1>      ;call arg2
4162      <1>      ; * name - 'u.namep' points to address of file/path name
4163      <1>      ;      in the user's program segment ('u.segmnt')
4164      <1>      ;      with offset in BX register (as sysopen argument 1).
4165      <1>      ; * argp - sysexec argument 2 is in CX register
4166      <1>      ;      which is on top of stack.
4167      <1>      ;
4168      <1>      ;      ; jsr r0,arg2 / arg0 in u.namep,arg1 on top of stack
4169      <1>      ;
4170      <1>      ; 23/06/2015 (32 bit modifications)
4171      <1>      ;
4172      0000D834 891D[C44E0100] <1>      mov     [u.namep], ebx ; argument 1
4173      <1>      ; 18/10/2015
4174      0000D83A 890D[944F0100] <1>      mov     [argv], ecx ; * ; argument 2
4175      0000D840 E86F020000 <1>      call    namei
4176      <1>      ;      ; jsr r0,namei / namei returns i-number of file
4177      <1>      ;      ; / named in sysexec call in r1
4178      <1>      ;jc     error
4179      <1>      ;      ; br error9
4180      0000D845 731E <1>      jnc     short sysexec_0
4181      <1>      ;
4182      <1>      ; 'file not found !' error
4183      0000D847 C705[104F0100]0C00- <1>      mov     dword [u.error], ERR_FILE_NOT_FOUND
4184      0000D84F 0000 <1>
4185      0000D851 E999EBFFFF <1>      jmp     error
4186      <1>      sysexec_not_exf:
4187      <1>      ; 'not executable file !' error
4188      0000D856 C705[104F0100]1600- <1>      mov     dword [u.error], ERR_NOT_EXECUTABLE
4189      0000D85E 0000 <1>
4190      0000D860 E98AEBFFFF <1>      jmp     error
4191      <1>      sysexec_0:
4192      0000D865 E854190000 <1>      call    iget
4193      <1>      ;      ; jsr r0,iget / get i-node for file to be executed
4194      0000D86A 66F705[464B0100]10- <1>      test     word [i.flgs], 10h
4195      0000D872 00 <1>
4196      <1>      ;      ; bit $20,i.flgs / is file executable
4197      <1>      ;
4198      0000D873 74E1 <1>      jz      short sysexec_not_exf
4199      <1>      ;jc     error
4200      <1>      ;      ; beq error9
4201      <1>      ;
4202      0000D875 E846190000 <1>      call    iopen
4203      <1>      ;      ; jsr r0,iopen / gets i-node for file with i-number
4204      <1>      ;      ; / given in r1 (opens file)
4205      <1>      ;
4206      <1>      ; AX = i-number of the file
4207      0000D87A 66F705[464B0100]20- <1>      test     word [i.flgs], 20h
4208      0000D882 00 <1>
4209      <1>      ;      ; bit $40,i.flgs / test user id on execution bit
4210      0000D883 7415 <1>      jz      short sysexec_1
4211      <1>      ;      ; beq 1f
4212      0000D885 803D[F84E0100]00 <1>      cmp     byte [u.uid], 0 ; 02/08/2013

```

```

4207          <1>          ; tstb u.uid / test user id
4208 0000D88C 760C          <1>          jna      short sysexec_1
4209          <1>          ; beq lf / super user
4210 0000D88E 8A0D[494B0100] <1>          mov     cl, [i.uid]
4211 0000D894 880D[F84E0100] <1>          mov     [u.uid], cl ; 02/08/2013
4212          <1>          ; movb i.uid,u.uid / put user id of owner of file
4213          <1>          ; / as process user id
4214          <1> sysexec_1:
4215          <1>          ; 18/10/2215
4216          <1>          ; 10/10/2015
4217          <1>          ; 24/07/2015
4218          <1>          ; 21/07/2015
4219          <1>          ; 25/06/2015
4220          <1>          ; 24/06/2015
4221          <1>          ; Moving arguments to the end of [u.upage]
4222          <1>          ; (by regarding page borders in user's memory space)
4223          <1>          ;
4224          <1>          ; 10/10/2015
4225          <1>          ; 21/07/2015
4226 0000D89A 89E5          <1>          mov     ebp, esp ; (**)
4227          <1>          ; 18/10/2015
4228 0000D89C 89EF          <1>          mov     edi, ebp
4229 0000D89E B900010000     <1>          mov     ecx, MAX_ARG_LEN ; 256
4230          <1>          ;sub     edi, MAX_ARG_LEN ; 256
4231 0000D8A3 29CF          <1>          sub     edi, ecx
4232 0000D8A5 89FC          <1>          mov     esp, edi
4233 0000D8A7 31C0          <1>          xor     eax, eax
4234 0000D8A9 A3[D44E0100]   <1>          mov     [u.nread], eax ; 0
4235 0000D8AE 49            <1>          dec     ecx ; 256 - 1
4236 0000D8AF 890D[D04E0100] <1>          mov     [u.count], ecx ; MAX_ARG_LEN - 1 ; 255
4237          <1>          ;mov     dword [u.count], MAX_ARG_LEN - 1 ; 255
4238          <1> sysexec_2:
4239 0000D8B5 8B35[944F0100] <1>          mov     esi, [argv] ; 18/10/2015
4240 0000D8BB E866000000     <1>          call    get_argp
4241 0000D8C0 B904000000     <1>          mov     ecx, 4 ; mov ecx, 4
4242          <1> sysexec_3:
4243          <1>          and     eax, eax
4244 0000D8C7 0F84E3070000     <1>          jz      sysexec_6
4245          <1>          ; 18/10/2015
4246 0000D8CD 010D[944F0100] <1>          add     [argv], ecx ; 4
4247 0000D8D3 66FF05[924F0100] <1>          inc     word [argc]
4248          <1>          ;
4249 0000D8DA A3[CC4E0100]   <1>          mov     [u.base], eax
4250          <1>          ; 23/10/2015
4251 0000D8DF 66C705[0C4F0100]00- <1>          mov     word [u.pcount], 0
4251 0000D8E7 00            <1>
4252          <1> sysexec_4:
4253 0000D8E8 E8830E0000     <1>          call    cpass ; get a character from user's core memory
4254 0000D8ED 750E          <1>          jnz     short sysexec_5
4255          <1>          ; (max. 255 chars + null)
4256          <1>          ; 18/10/2015
4257 0000D8EF 28C0          <1>          sub     al, al
4258 0000D8F1 AA            <1>          stosb
4259 0000D8F2 FF05[D44E0100] <1>          inc     dword [u.nread]
4260 0000D8F8 E9B3070000     <1>          jmp     sysexec_6 ; 24/04/2016
4261          <1> sysexec_5:
4262 0000D8FD AA            <1>          stosb
4263 0000D8FE 20C0          <1>          and     al, al
4264 0000D900 75E6          <1>          jnz     short sysexec_4
4265 0000D902 B904000000     <1>          mov     ecx, 4
4266 0000D907 390D[904F0100] <1>          cmp     [ncount], ecx ; 4
4267 0000D90D 72A6          <1>          jb      short sysexec_2
4268 0000D90F 8B35[8C4F0100] <1>          mov     esi, [nbase]
4269 0000D915 010D[8C4F0100] <1>          add     [nbase], ecx ; 4
4270 0000D91B 66290D[904F0100] <1>          sub     [ncount], cx
4271 0000D922 8B06          <1>          mov     eax, [esi]
4272 0000D924 EB9F          <1>          jmp     short sysexec_3
4273          <1>
4274          <1> get_argp:
4275          <1>          ; 18/10/2015 (nbase, ncount)
4276          <1>          ; 21/07/2015
4277          <1>          ; 24/06/2015 (Retro UNIX 386 v1)
4278          <1>          ; Get (virtual) address of argument from user's core memory
4279          <1>          ;
4280          <1>          ; INPUT:
4281          <1>          ;     esi = virtual address of argument pointer
4282          <1>          ; OUTPUT:
4283          <1>          ;     eax = virtual address of argument
4284          <1>          ;
4285          <1>          ; Modified registers: EAX, EBX, ECX, EDX, ESI
4286          <1>          ;
4287 0000D926 833D[044F0100]00 <1>          cmp     dword [u.ppgdir], 0 ; /etc/init ?
4288          <1>          ; (the caller is kernel)
4289 0000D92D 7667          <1>          jna     short get_argpk
4290          <1>          ;
4291 0000D92F 89F3          <1>          mov     ebx, esi
4292 0000D931 E85879FFFF     <1>          call    get_physical_addr ; get physical address
4293 0000D936 0F8289000000     <1>          jc      get_argp_err
4294 0000D93C A3[8C4F0100]   <1>          mov     [nbase], eax ; physical address
4295 0000D941 66890D[904F0100] <1>          mov     [ncount], cx ; remain byte count in page (1-4096)
4296 0000D948 B804000000     <1>          mov     eax, 4 ; 21/07/2015
4297 0000D94D 6639C1          <1>          cmp     cx, ax ; 4
4298 0000D950 735D          <1>          jnb     short get_argp2
4299 0000D952 89F3          <1>          mov     ebx, esi
4300 0000D954 01CB          <1>          add     ebx, ecx
4301 0000D956 E83379FFFF     <1>          call    get_physical_addr ; get physical address
4302 0000D95B 7268          <1>          jc      short get_argp_err
4303          <1>          ;push     esi
4304 0000D95D 89C6          <1>          mov     esi, eax
4305 0000D95F 66870D[904F0100] <1>          xchg     cx, [ncount]
4306 0000D966 8735[8C4F0100] <1>          xchg     esi, [nbase]
4307 0000D96C B504          <1>          mov     ch, 4

```

```

4308 0000D96E 28CD      <1>      sub    ch, cl
4309                  <1> get_argp0:
4310 0000D970 AC        <1>      lodsb
4311 0000D971 6650      <1>      push   ax
4312 0000D973 FEC9      <1>      dec    cl
4313 0000D975 75F9      <1>      jnz     short get_argp0
4314 0000D977 8B35[8C4F0100] <1>      mov     esi, [nbase]
4315                  <1>      ; 21/07/2015
4316 0000D97D 0FB6C5    <1>      movzx   eax, ch
4317 0000D980 0105[8C4F0100] <1>      add     [nbase], eax
4318 0000D986 662905[904F0100] <1>      sub     [ncount], ax
4319                  <1> get_argp1:
4320 0000D98D AC        <1>      lodsb
4321 0000D98E FECD      <1>      dec     ch
4322 0000D990 743D      <1>      jz      short get_argp3
4323 0000D992 6650      <1>      pushax
4324 0000D994 EBF7      <1>      jmp     short get_argp1
4325                  <1> get_argpk:
4326                  <1>      ; Argument is in kernel's memory space
4327 0000D996 66C705[904F0100]00- <1>      mov     word [ncount], PAGE_SIZE ; 4096
4327 0000D99E 10        <1>
4328 0000D99F 8935[8C4F0100] <1>      mov     [nbase], esi
4329 0000D9A5 8305[8C4F0100]04 <1>      add     dword [nbase], 4
4330 0000D9AC 8B06      <1>      mov     eax, [esi] ; virtual addr. = physcal addr.
4331 0000D9AE C3        <1>      retn
4332                  <1> get_argp2:
4333                  <1>      ; 21/07/2015
4334                  <1>      ;mov     eax, 4
4335 0000D9AF 8B15[8C4F0100] <1>      mov     edx, [nbase] ; 18/10/2015
4336 0000D9B5 0105[8C4F0100] <1>      add     [nbase], eax
4337 0000D9BB 662905[904F0100] <1>      sub     [ncount], ax
4338                  <1>      ;
4339 0000D9C2 8B02      <1>      mov     eax, [edx]
4340 0000D9C4 C3        <1>      retn
4341                  <1> get_argp_err:
4342 0000D9C5 A3[104F0100] <1>      mov     [u.error], eax
4343 0000D9CA E920EAF7FF <1>      jmp     error
4344                  <1> get_argp3:
4345 0000D9CF B103      <1>      mov     cl, 3
4346                  <1> get_argp4:
4347 0000D9D1 C1E008    <1>      shl     eax, 8
4348 0000D9D4 665A      <1>      pop     dx
4349 0000D9D6 88D0      <1>      mov     al, dl
4350 0000D9D8 E2F7      <1>      loop    get_argp4
4351                  <1>      ;pop     esi
4352 0000D9DA C3        <1>      retn
4353                  <1>
4354                  <1> sysstat:
4355                  <1>      ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
4356                  <1>      ; temporary !
4357 0000D9DB B801000000 <1>      mov     eax, ERR_INV_FNUMBER ; 'invalid function number !'
4358 0000D9E0 A3[104F0100] <1>      mov     [u.error], eax
4359 0000D9E5 A3[AC4E0100] <1>      mov     [u.r0], eax
4360 0000D9EA E900EAF7FF <1>      jmp     error
4361                  <1>
4362                  <1> sysfstat:
4363                  <1>      ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
4364                  <1>      ; temporary !
4365 0000D9EF B801000000 <1>      mov     eax, ERR_INV_FNUMBER ; 'invalid function number !'
4366 0000D9F4 A3[104F0100] <1>      mov     [u.error], eax
4367 0000D9F9 A3[AC4E0100] <1>      mov     [u.r0], eax
4368 0000D9FE E9ECE9FFFF <1>      jmp     error
4369                  <1>
4370                  <1> fclose:
4371                  <1>      ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
4372                  <1>      ;
4373                  <1>      ; 18/06/2015 (Retro UNIX 386 v1 - Beginning)
4374                  <1>      ; (32 bit offset pointer modification)
4375                  <1>      ; 19/04/2013 - 12/01/2014 (Retro UNIX 8086 v1)
4376                  <1>      ;
4377                  <1>      ; Given the file descriptor (index to the u.fp list)
4378                  <1>      ; 'fclose' first gets the i-number of the file via 'getf'.
4379                  <1>      ; If i-node is active (i-number > 0) the entry in
4380                  <1>      ; u.fp list is cleared. If all the processes that opened
4381                  <1>      ; that file close it, then fsp etry is freed and the file
4382                  <1>      ; is closed. If not a return is taken.
4383                  <1>      ; If the file has been deleted while open, 'anyi' is called
4384                  <1>      ; to see anyone else has it open, i.e., see if it is appears
4385                  <1>      ; in another entry in the fsp table. Upon return from 'anyi'
4386                  <1>      ; a check is made to see if the file is special.
4387                  <1>      ;
4388                  <1>      ; INPUTS ->
4389                  <1>      ;   r1 - contains the file descriptor (value=0,1,2...)
4390                  <1>      ;   u.fp - list of entries in the fsp table
4391                  <1>      ;   fsp - table of entries (4 words/entry) of open files.
4392                  <1>      ; OUTPUTS ->
4393                  <1>      ;   r1 - contains the same file descriptor
4394                  <1>      ;   r2 - contains i-number
4395                  <1>      ;
4396                  <1>      ; ((AX = R1))
4397                  <1>      ; ((Modified registers: EDX, EBX, ECX, ESI, EDI, EBP))
4398                  <1>      ;
4399                  <1>      ; Retro UNIX 8086 v1 modification : CF = 1
4400                  <1>      ; if i-number of the file is 0. (error)
4401                  <1>      ;
4402                  <1>      ; TRDOS 386 (06/10/2016)
4403                  <1>      ;
4404                  <1>      ; INPUT:
4405                  <1>      ;   EAX = File Handle (File Descriptor, File Index)
4406                  <1>      ;
4407                  <1>      ; OUTPUT:
4408                  <1>      ;   CF = 1 -> File not open !

```

```

4409      <1>      ;      CF = 0 -> OK!
4410      <1>      ;      EBX = File Number (System)
4411      <1>      ;      [cdev] = Logical DOS Drive Number
4412      <1>      ;      EAX = File Handle/Number (user)
4413      <1>      ;
4414      <1>      ; Modified Registers: EBX
4415      <1>
4416 0000DA03 50      <1>      push    eax ; File handle
4417      <1>
4418 0000DA04 E846000000      <1>      call   getf
4419 0000DA09 0F827D200000      <1>      jc     device_close ; eax = device number
4420      <1>
4421 0000DA0F 80BB[4A490100]01      <1>      cmp     byte [ebx+OF_MODE], 1 ; open mode ; 0 = empty entry
4422 0000DA16 722E      <1>      jnb     short fclose_1 ; 1 = read, 2 = write
4423      <1>
4424 0000DA18 83F801      <1>      cmp     eax, 1 ; is the first cluster number > 0
4425 0000DA1B 7229      <1>      jnb     short fclose_1 ; no, this is empty entry
4426      <1>
4427      <1> fclose_0:
4428 0000DA1D FE8B[5E490100]      <1>      dec     byte [ebx+OF_OPENCOUNT] ; decrement the number of processes
4429      <1> ; that have opened the file
4430 0000DA23 7921      <1>      jns     short fclose_1 ; jump if not negative (jump if bit 7 is 0)
4431      <1> ; if all processes haven't closed the file, return
4432      <1>
4433      <1> ; eax ; First cluster
4434 0000DA25 31C0      <1>      xor     eax, eax ; 0
4435 0000DA27 8883[4A490100]      <1>      mov     [ebx+OF_MODE], al ; 0 = empty entry
4436      <1> ;mov [ebx+OF_STATUS], al ; 0 = empty entry
4437 0000DA2D 66C1E302      <1>      shl     bx, 2
4438 0000DA31 8983[18490100]      <1>      mov     [ebx+OF_FCLUSTER], eax ; 0
4439 0000DA37 8983[304A0100]      <1>      mov     [ebx+OF_CCLUSTER], eax ; 0
4440      <1> ;mov [ebx+OF_CCINDEX], eax ; 0
4441 0000DA3D A3[BC4E0100]      <1>      mov     [u.fofp], eax ; 0
4442 0000DA42 66C1EB02      <1>      shr     bx, 2
4443      <1> fclose_1: ; 1:
4444 0000DA46 58      <1>      pop     eax ; File handle (File Descriptor, File Index)
4445 0000DA47 C680[B24E0100]00      <1>      mov     byte [eax+u.fp], 0 ; clear that entry in the u.fp list
4446 0000DA4E C3      <1>      retn
4447      <1>
4448      <1> getf:
4449      <1> ; 12/10/2016
4450      <1> ; 11/10/2016
4451      <1> ; 08/10/2016
4452      <1> ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
4453      <1> ; / get the device number and the i-number of an open file
4454      <1> ; 13/05/2015
4455      <1> ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
4456      <1> ; 19/04/2013 - 18/11/2013 (Retro UNIX 8086 v1)
4457      <1> ;
4458 0000DA4F 89C3      <1>      mov     ebx, eax
4459      <1> getf1:
4460 0000DA51 83FB0A      <1>      cmp     ebx, 10
4461 0000DA54 730A      <1>      jnb     short getf2
4462 0000DA56 8A9B[B24E0100]      <1>      mov     bl, [ebx+u.fp]
4463 0000DA5C 08DB      <1>      or      bl, bl
4464 0000DA5E 7503      <1>      jnz     short getf3
4465      <1> getf2:
4466      <1> ; 'File not open !' error (ax=0)
4467 0000DA60 29C0      <1>      sub     eax, eax
4468 0000DA62 C3      <1>      retn
4469      <1> getf3:
4470 0000DA63 F6C380      <1>      test    bl, 80h
4471 0000DA66 7530      <1>      jnz     short getf5 ; device
4472 0000DA68 FECB      <1>      dec     bl ; 0 based
4473 0000DA6A 8A83[40490100]      <1>      mov     al, [ebx+OF_DRIVE]
4474 0000DA70 A2[8C4E0100]      <1>      mov     [cdev], al
4475 0000DA75 C0E302      <1>      shl     bl, 2 ; *4 (dword offset)
4476 0000DA78 8B83[90490100]      <1>      mov     eax, [ebx+OF_SIZE]
4477 0000DA7E A3[9D4F0100]      <1>      mov     [i.size], eax ; file size
4478 0000DA83 8D83[68490100]      <1>      lea     eax, [ebx+OF_POINTER] ;12/10/2016
4479 0000DA89 A3[BC4E0100]      <1>      mov     [u.fofp], eax
4480 0000DA8E 8B83[18490100]      <1>      mov     eax, [ebx+OF_FCLUSTER]
4481 0000DA94 C0EB02      <1>      shr     bl, 2 ; /4 (byte offset)
4482      <1> getf4:
4483 0000DA97 C3      <1>      retn
4484      <1> getf5:
4485      <1> ; get device number
4486 0000DA98 80E37F      <1>      and     bl, 7Fh ; 1 to 7Fh
4487 0000DA9B FECB      <1>      dec     bl ; 0 based (0 to 7Eh)
4488 0000DA9D 8A83[72470100]      <1>      mov     al, [ebx+DEV_DRIVER]
4489 0000DAA3 8AAB[DC460100]      <1>      mov     ch, [ebx+DEV_ACCESS]
4490 0000DAA9 8A8B[90470100]      <1>      mov     cl, [ebx+DEV_OPENMODE]
4491 0000DAAF 80E5FE      <1>      and     ch, 0FEh ; reset bit 0 ; dev_close
4492 0000DAB2 F9      <1>      stc     ; cf = 1
4493 0000DAB3 C3      <1>      retn
4494      <1>
4495      <1> namei:
4496      <1> ; 04/12/2015 (14 byte file names)
4497      <1> ; 18/10/2015 (nbase, ncount)
4498      <1> ; 12/10/2015
4499      <1> ; 21/08/2015
4500      <1> ; 18/07/2015
4501      <1> ; 02/07/2015
4502      <1> ; 17/06/2015
4503      <1> ; 16/06/2015 (Retro UNIX 386 v1 - Beginning)
4504      <1> ; 24/04/2013 - 31/07/2013 (Retro UNIX 8086 v1)
4505      <1> ;
4506      <1> ; 'namei' takes a file path name and returns i-number of
4507      <1> ; the file in the current directory or the root directory
4508      <1> ; (if the first character of the pathname is '/').
4509      <1> ;
4510      <1> ; INPUTS ->

```



```

4511      <1>      ; u.namep - points to a file path name
4512      <1>      ; u.cdir - i-number of users directory
4513      <1>      ; u.cdev - device number on which user directory resides
4514      <1>      ; OUTPUTS ->
4515      <1>      ; r1 - i-number of file
4516      <1>      ; cdev
4517      <1>      ; u.dirbuf - points to directory entry where a match
4518      <1>      ; occurs in the search for file path name.
4519      <1>      ; If no match u.dirb points to the end of
4520      <1>      ; the directory and r1 = i-number of the current
4521      <1>      ; directory.
4522      <1>      ; ((AX = R1))
4523      <1>      ;
4524      <1>      ; (Retro UNIX Prototype : 07/10/2012 - 05/01/2013, UNIXCOPY.ASM)
4525      <1>      ; ((Modified registers: EDX, EBX, ECX, ESI, EDI, EBP))
4526      <1>      ;
4527      <1>
4528 0000DAB4 66A1[B04E0100] <1>      mov     ax, [u.cdir]
4529      <1>      ; mov u.cdir,r1 / put the i-number of current directory
4530      <1>      ; / in r1
4531 0000DABA 668B15[F64E0100] <1>      mov     dx, [u.cdrv]
4532 0000DAC1 668915[8C4E0100] <1>      mov     [cdev], dx      ; NOTE: Retro UNIX 8086 v1
4533      <1>      ; device/drive number is in 1 byte,
4534      <1>      ; not in 1 word!
4535      <1>      ; mov u.cdev,cdev / device number for users directory
4536      <1>      ; / into cdev
4537      <1>      ; 12/10/2015
4538      <1>      ; 16/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
4539      <1>      ; convert virtual (pathname) addr to physical address
4540 0000DAC8 E82C010000 <1>      call    trans_addr_nmbp ; 12/10/2015
4541      <1>      ; esi = physical address of [u.namep]
4542      <1>      ; ecx = byte count in the page
4543 0000DACD 803E2F <1>      cmp     byte [esi], '/'
4544      <1>      ; cmpb *u.namep,$'/ / is first char in file name a /
4545 0000DAD0 751E <1>      jne     short namei_1
4546      <1>      ; bne lf
4547 0000DAD2 FF05[C44E0100] <1>      inc     dword [u.namep]
4548      <1>      ; inc u.namep / go to next char
4549 0000DAD8 6649 <1>      dec     cx ; remain byte count in the page
4550 0000DADA 7506 <1>      jnz     short namei_0
4551      <1>      ; 12/10/2015
4552 0000DADC E818010000 <1>      call    trans_addr_nmbp ; convert virtual address to physical
4553      <1>      ; esi = physical address (page start + offset)
4554      <1>      ; ecx = byte count in the page
4555 0000DAE1 4E <1>      dec     esi
4556      <1>      namei_0:
4557 0000DAE2 46 <1>      inc     esi ; go to next char
4558 0000DAE3 66A1[964E0100] <1>      mov     ax, [rootdir] ; 09/07/2013
4559      <1>      ; mov rootdir,r1 / put i-number of rootdirectory in r1
4560 0000DAE9 C605[8C4E0100]00 <1>      mov     byte [cdev], 0
4561      <1>      ; clr cdev / clear device number
4562      <1>      namei_1: ; 1:
4563 0000DAF0 F606FF <1>      test    byte [esi], 0FFh
4564 0000DAF3 74A2 <1>      jz      short getf4
4565      <1>      ; jz      nig
4566      <1>      ; tstb *u.namep / is the character in file name a nul
4567      <1>      ; beq nig / yes, end of file name reached;
4568      <1>      ; / branch to "nig"
4569      <1>      namei_2: ; 1:
4570      <1>      ; 18/10/2015
4571 0000DAF5 8935[8C4F0100] <1>      mov     [nbase], esi
4572 0000DAFB 66890D[904F0100] <1>      mov     [ncount], cx
4573      <1>      ;
4574      <1>      ;mov     dx, 2
4575 0000DB02 B202 <1>      mov     dl, 2 ; user flag (read, non-owner)
4576 0000DB04 E8BB160000 <1>      call    access
4577      <1>      ; jsr r0,access; 2 / get i-node with i-number r1
4578      <1>      ; 'access' will not return here if user has not "r" permission !
4579 0000DB09 66F705[464B0100]00- <1>      test    word [i.flgs], 4000h
4579 0000DB11 40 <1>
4580      <1>      ; bit $40000,i.flgs / directory i-node?
4581 0000DB12 746A <1>      jz      short namei_err
4582      <1>      ; beq error3 / no, got an error
4583      <1>      ; 16/06/2015 - 32 bit modifications (Retro UNIX 386 v1)
4584 0000DB14 31C0 <1>      xor     eax, eax
4585 0000DB16 A3[C84E0100] <1>      mov     [u.off], eax ; 0
4586 0000DB1B 66A1[9D4F0100] <1>      mov     ax, [i.size]
4587 0000DB21 A3[C04E0100] <1>      mov     [u.dirp], eax
4588      <1>      ; mov i.size,u.dirp / put size of directory in u.dirp
4589      <1>      ; clr u.off / u.off is file offset used by user
4590 0000DB26 C705[BC4E0100]- <1>      mov     dword [u.fofp], u.off
4590 0000DB2C [C84E0100] <1>
4591      <1>      ; mov $u.off,u.fofp / u.fofp is a pointer to
4592      <1>      ; / the offset portion of fsp entry
4593      <1>      namei_3: ; 2:
4594 0000DB30 C705[CC4E0100]- <1>      mov     dword [u.base], u.dirbuf
4594 0000DB36 [E04E0100] <1>
4595      <1>      ; mov $u.dirbuf,u.base / u.dirbuf holds a file name
4596      <1>      ; / copied from a directory
4597 0000DB3A C705[D04E0100]1000- <1>      mov     dword [u.count], 16 ; 04/12/2015 (10 -> 16)
4597 0000DB42 0000 <1>
4598      <1>      ; mov $10.,u.count / u.count is byte count
4599      <1>      ; / for reads and writes
4600 0000DB44 66A1[994F0100] <1>      mov     ax, [ii]
4601      <1>      ; 31/07/2013 ('namei_r') - 16/06/2015 ('u.kcall')
4602 0000DB4A FE05[0E4F0100] <1>      inc     byte [u.kcall] ; the caller is 'namei' sign
4603 0000DB50 E8A2070000 <1>      call    readi
4604      <1>      ; jsr r0,readi / read 10. bytes of file
4605      <1>      ; with i-number (r1); i.e. read a directory entry
4606 0000DB55 8B0D[D44E0100] <1>      mov     ecx, [u.nread]
4607 0000DB5B 09C9 <1>      or      ecx, ecx
4608      <1>      ; tst u.nread

```

```

4609 0000DB5D 741B      <1>      jz      short nib
4610                    <1>      ; ble nib / gives error return
4611                    <1>      ;
4612 0000DB5F 668B1D[E04E0100] <1>      mov     bx, [u.dirbuf]
4613 0000DB66 6621DB      <1>      and     bx, bx
4614                    <1>      ; tst u.dirbuf /
4615 0000DB69 7522      <1>      jnz     short namei_4
4616                    <1>      ; bne 3f / branch when active directory entry
4617                    <1>      ; / (i-node word in entry non zero)
4618 0000DB6B A1[C84E0100] <1>      mov     eax, [u.off]
4619 0000DB70 83E810      <1>      sub     eax, 16 ; 04/12/2015 (10 -> 16)
4620 0000DB73 A3[C04E0100] <1>      mov     [u.dirp], eax
4621                    <1>      ; mov u.off,u.dirp
4622                    <1>      ; sub $10.,u.dirp
4623 0000DB78 EBB6      <1>      jmp     short namei_3
4624                    <1>      ; br 2b
4625                    <1>
4626                    <1>      ; 18/07/2013
4627                    <1> nib:
4628 0000DB7A 31C0      <1>      xor     eax, eax ; xor ax, ax ; ax = 0 -> file not found
4629 0000DB7C F9        <1>      stc
4630                    <1> nig:
4631 0000DB7D C3        <1>      retn
4632                    <1>
4633                    <1> namei_err:
4634                    <1>      ; 16/06/2015
4635 0000DB7E C705[104F0100]1300- <1>      mov     dword [u.error], ERR_NOT_DIR ; 'not a directory !' error
4635 0000DB86 0000      <1>
4636 0000DB88 E962E8FFFF <1>      jmp     error
4637                    <1>
4638                    <1> namei_4: ; 3:
4639                    <1>      ; 18/10/2015
4640                    <1>      ; 12/10/2015
4641                    <1>      ; 21/08/2015
4642                    <1>      ; 18/07/2015
4643 0000DB8D 8B2D[C44E0100] <1>      mov     ebp, [u.namep]
4644                    <1>      ; mov u.namep,r2 / u.namep points into a file name string
4645 0000DB93 BF[E24E0100] <1>      mov     edi, u.dirbuf + 2
4646                    <1>      ; mov $u.dirbuf+2,r3 / points to file name of directory entry
4647                    <1>      ; 18/10/2015
4648 0000DB98 8B35[8C4F0100] <1>      mov     esi, [nbase]
4649 0000DB9E 668B0D[904F0100] <1>      mov     cx, [ncount]
4650                    <1>      ;
4651 0000DBA5 6621C9      <1>      and     cx, cx
4652 0000DBA8 7505      <1>      jnz     short namei_5
4653                    <1>      ;
4654 0000DBAA E850000000 <1>      call    trans_addr_nm ; convert virtual address to physical
4655                    <1>      ; esi = physical address (page start + offset)
4656                    <1>      ; ecx = byte count in the page
4657                    <1> namei_5: ; 3:
4658                    <1>      inc     ebp ; 18/07/2015
4659 0000DBB0 AC        <1>      lodsb   ; mov al, [esi] ; inc esi (al = r4)
4660                    <1>      ; movb (r2)+,r4 / move a character from u.namep string into r4
4661 0000DBB1 08C0      <1>      or      al, al
4662 0000DBB3 741D      <1>      jz      short namei_7
4663                    <1>      ; beq 3f / if char is nul, then the last char in string
4664                    <1>      ; / has been moved
4665 0000DBB5 3C2F      <1>      cmp     al, '/'
4666                    <1>      ; cmp r4,$'/ / is char a </>
4667 0000DBB7 7419      <1>      je      short namei_7
4668                    <1>      ; beq 3f
4669                    <1>      ; 12/10/2015
4670 0000DBB9 6649      <1>      dec     cx ; remain byte count in the page
4671 0000DBBB 7505      <1>      jnz     short namei_6
4672 0000DBBD E83D000000 <1>      call    trans_addr_nm ; convert virtual address to physical
4673                    <1>      ; esi = physical address (page start + offset)
4674                    <1>      ; ecx = byte count in the page
4675                    <1> namei_6:
4676 0000DBC2 81FF[F04E0100] <1>      cmp     edi, u.dirbuf + 16 ; 04/12/2015 (10 -> 16)
4677                    <1>      ; cmp r3,$u.dirbuf+10. / have I checked
4678                    <1>      ; / all 8 bytes of file name
4679 0000DBC8 74E5      <1>      je      short namei_5
4680                    <1>      ; beq 3b
4681 0000DBCA AE        <1>      scasb
4682                    <1>      ; cmpb (r3)+,r4 / compare char in u.namep string to file name
4683                    <1>      ; / char read from directory
4684 0000DBCB 74E2      <1>      je      short namei_5
4685                    <1>      ; beq 3b / branch if chars match
4686                    <1>
4687 0000DBCD E95EFFFFFF <1>      jmp     namei_3 ; 2b
4688                    <1>      ; br 2b / file names do not match go to next directory entry
4689                    <1> namei_7: ; 3:
4690 0000DBD2 81FF[F04E0100] <1>      cmp     edi, u.dirbuf + 16 ; 04/12/2015 (10 -> 16)
4691                    <1>      ; cmp r3,$u.dirbuf+10. / if equal all 8 bytes were matched
4692 0000DBD8 740A      <1>      je      short namei_8
4693                    <1>      ; beq 3f
4694 0000DBDA 8A27      <1>      mov     ah, [edi]
4695                    <1>      ;inc edi
4696 0000DBDC 20E4      <1>      and     ah, ah
4697                    <1>      ; tstb (r3)+ /
4698 0000DBDE 0F854CFFFFFF <1>      jnz     namei_3
4699                    <1>      ; bne 2b
4700                    <1> namei_8: ; 3
4701 0000DBE4 892D[C44E0100] <1>      mov     [u.namep], ebp ; 18/07/2015
4702                    <1>      ; mov r2,u.namep / u.namep points to char
4703                    <1>      ; / following a / or nul
4704                    <1> ;mov bx, [u.dirbuf]
4705                    <1>      ; mov u.dirbuf,r1 / move i-node number in directory
4706                    <1>      ; / entry to r1
4707 0000DBEA 20C0      <1>      and     al, al
4708                    <1>      ; tst r4 / if r4 = 0 the end of file name reached,
4709                    <1>      ; / if r4 = </> then go to next directory

```

```

4710      <1>      ; mov ax, bx
4711 0000DBEC 66A1[E04E0100] <1>      mov ax, [u.dirbuf] ; 17/06/2015
4712 0000DBF2 0F85FDFEFFFF <1>      jnz namei_2
4713      <1>      ; bne 1b
4714      <1>      ; AX = i-number of the file
4715      <1> ;;nig:
4716 0000DBF8 C3 <1>      retn
4717      <1>      ; tst (r0)+ / gives non-error return
4718      <1> ;;nib:
4719      <1> ;; xor ax, ax ; Retro UNIX 8086 v1 modification !
4720      <1>      ; ax = 0 -> file not found
4721      <1> ;; stc ; 27/05/2013
4722      <1> ;; retn
4723      <1>      ; rts r0
4724      <1>
4725      <1> trans_addr_nmbp:
4726      <1>      ; 18/10/2015
4727      <1>      ; 12/10/2015
4728 0000DBF9 8B2D[C44E0100] <1>      mov ebp, [u.namep]
4729      <1> trans_addr_nm:
4730      <1>      ; Convert virtual (pathname) address to physical address
4731      <1>      ; (Retro UNIX 386 v1 feature only !)
4732      <1>      ; 18/10/2015
4733      <1>      ; 12/10/2015 (u.pnbase & u.pncount has been removed from code)
4734      <1>      ; 02/07/2015
4735      <1>      ; 17/06/2015
4736      <1>      ; 16/06/2015
4737      <1>      ;
4738      <1>      ; INPUTS:
4739      <1>      ; ebp = pathname address (virtual) ; [u.namep]
4740      <1>      ; [u.pgdir] = user's page directory
4741      <1>      ; OUTPUT:
4742      <1>      ; esi = physical address of the pathname
4743      <1>      ; ecx = remain byte count in the page
4744      <1>      ;
4745      <1>      ; (Modified registers: EAX, EBX, ECX, EDX, ESI)
4746      <1>      ;
4747 0000DBFF 833D[044F0100]00 <1>      cmp dword [u.ppgdir], 0 ; /etc/init ? (sysexec)
4748 0000DC06 7618 <1>      jna short trans_addr_nmk ; the caller is os kernel;
4749      <1>      ; it is already physical address
4750      <1>      push eax
4751 0000DC09 89EB <1>      mov ebx, ebp ; [u.namep] ; pathname address (virtual)
4752 0000DC0B E87E76FFFF <1>      call get_physical_addr ; get physical address
4753 0000DC10 7204 <1>      jc short tr_addr_nm_err
4754      <1>      ; 18/10/2015
4755      <1>      ; eax = physical address
4756      <1>      ; cx = remain byte count in page (1-4096)
4757      <1>      ; 12/10/2015 (cx = [u.pncount])
4758 0000DC12 89C6 <1>      mov esi, eax ; 12/10/2015 (esi=[u.pnbase])
4759 0000DC14 58 <1>      pop eax
4760 0000DC15 C3 <1>      retn
4761      <1>
4762      <1> tr_addr_nm_err:
4763 0000DC16 A3[104F0100] <1>      mov [u.error], eax
4764      <1>      ;pop eax
4765 0000DC1B E9CFE7FFFF <1>      jmp error
4766      <1>
4767      <1> trans_addr_nmk:
4768      <1>      ; 12/10/2015
4769      <1>      ; 02/07/2015
4770 0000DC20 8B35[C44E0100] <1>      mov esi, [u.namep] ; [u.pnbase]
4771 0000DC26 66B90010 <1>      mov cx, PAGE_SIZE ; 4096 ; [u.pncount]
4772 0000DC2A C3 <1>      retn
4773      <1>
4774      <1> syschdir:
4775      <1>      ; / makes the directory specified in the argument
4776      <1>      ; / the current directory
4777      <1>      ;
4778      <1>      ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
4779      <1>      ; 19/06/2013 (Retro UNIX 8086 v1)
4780      <1>      ;
4781      <1>      ; 'syschdir' makes the directory specified in its argument
4782      <1>      ; the current working directory.
4783      <1>      ;
4784      <1>      ; Calling sequence:
4785      <1>      ; syschdir; name
4786      <1>      ; Arguments:
4787      <1>      ; name - address of the path name of a directory
4788      <1>      ; terminated by nul byte.
4789      <1>      ; Inputs: -
4790      <1>      ; Outputs: -
4791      <1>      ; .....
4792      <1>      ;
4793      <1>      ; Retro UNIX 8086 v1 modification:
4794      <1>      ; The user/application program puts address of
4795      <1>      ; the path name in BX register as 'syschdir'
4796      <1>      ; system call argument.
4797      <1>
4798 0000DC2B 891D[C44E0100] <1>      mov [u.namep], ebx
4799      <1>      ;jsr r0,arg; u.namep / u.namep points to path name
4800 0000DC31 E87EFEFFFF <1>      call namei
4801      <1>      ; jsr r0,namei / find its i-number
4802      <1>      ;jc error
4803      <1>      ; br error3
4804 0000DC36 730F <1>      jnc short syschdir0
4805      <1>      ; 'directory not found !' error
4806 0000DC38 C705[104F0100]0C00- <1>      mov dword [u.error], ERR_DIR_NOT_FOUND ; 12
4807 0000DC40 0000 <1>
4808 0000DC42 E9A8E7FFFF <1>      jmp error
4809      <1> syschdir0:
4810      <1>      call access
4811      <1>      ; jsr r0,access; 2 / get i-node into core

```

```

4811 0000DC4C 66F705[464B0100]00- <1> test word [i.flgs], 4000h
4811 0000DC54 40 <1>
4812 <1> ; bit $40000,i.flgs / is it a directory?
4813 <1> ;jz error
4814 <1> ; beq error3 / no error
4815 0000DC55 750F <1> jnz short syschdir1
4816 0000DC57 C705[104F0100]1300- <1> mov dword [u.error], ERR_NOT_DIR ; 'not a valid directory !'
4816 0000DC5F 0000 <1>
4817 0000DC61 E989E7FFFF <1> jmp error
4818 <1> syschdir1:
4819 0000DC66 66A3[B04E0100] <1> mov [u.cdir], ax
4820 <1> ; mov r1,u.cdir / move i-number to users
4821 <1> ; / current directory
4822 0000DC6C 66A1[8C4E0100] <1> mov ax, [cdev]
4823 0000DC72 66A3[F64E0100] <1> mov [u.cdrv], ax
4824 <1> ; mov cdev,u.cdev / move its device to users
4825 <1> ; / current device
4826 0000DC78 E992E7FFFF <1> jmp sysret
4827 <1> ; br sysret3
4828 <1>
4829 <1> syschmod: ; < change mode of file >
4830 <1> ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
4831 <1> ; temporary !
4832 0000DC7D B801000000 <1> mov eax, ERR_INV_FNUMBER ; 'invalid function number !'
4833 0000DC82 A3[104F0100] <1> mov [u.error], eax
4834 0000DC87 A3[AC4E0100] <1> mov [u.r0], eax
4835 0000DC8C E95EE7FFFF <1> jmp error
4836 <1>
4837 <1> isown:
4838 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
4839 <1> ; 04/05/2013 - 07/07/2013 (Retro UNIX 8086 v1)
4840 <1> ;
4841 <1> ; 'isown' is given a file name (the 1st argument).
4842 <1> ; It find the i-number of that file via 'namei'
4843 <1> ; then gets the i-node into core via 'iget'.
4844 <1> ; It then tests to see if the user is super user.
4845 <1> ; If not, it cheks to see if the user is owner of
4846 <1> ; the file. If he is not an error occurs.
4847 <1> ; If user is the owner 'setimod' is called to indicate
4848 <1> ; the inode has been modified and the 2nd argument of
4849 <1> ; the call is put in r2.
4850 <1> ;
4851 <1> ; INPUTS ->
4852 <1> ; arguments of syschmod and syschown calls
4853 <1> ; OUTPUTS ->
4854 <1> ; u.uid - id of user
4855 <1> ; imod - set to a 1
4856 <1> ; r2 - contains second argument of the system call
4857 <1> ;
4858 <1> ; ((AX=R2) output as 2nd argument)
4859 <1> ;
4860 <1> ; ((Modified registers: eAX, eDX, eBX, eCX, eSI, eDI, eBP))
4861 <1> ;
4862 <1> ; jsr r0,arg2 / u.namep points to file name
4863 <1> ;; ! 2nd argument on top of stack !
4864 <1> ;; 22/06/2015 - 32 bit modifications
4865 <1> ;; 07/07/2013
4866 0000DC91 891D[C44E0100] <1> mov [u.namep], ebx ;; 1st argument
4867 0000DC97 51 <1> push ecx ;; 2nd argument
4868 <1> ;;
4869 0000DC98 E817FEFFFF <1> call namei
4870 <1> ; jsr r0,namei / get its i-number
4871 <1> ; Retro UNIX 8086 v1 modification !
4872 <1> ; ax = 0 -> file not found
4873 <1> ;and ax, ax
4874 <1> ;jz error
4875 <1> ;jc error ; 27/05/2013
4876 <1> ; br error3
4877 0000DC9D 730F <1> jnc short isown0
4878 <1> ; 'file not found !' error
4879 0000DC9F C705[104F0100]0C00- <1> mov dword [u.error], ERR_FILE_NOT_FOUND ; 12
4879 0000DCA7 0000 <1>
4880 0000DCA9 E941E7FFFF <1> jmp error
4881 <1> isown0:
4882 0000DCAE E80B150000 <1> call iget
4883 <1> ; jsr r0,iget / get i-node into core
4884 0000DCB3 A0[F84E0100] <1> mov al, [u.uid] ; 02/08/2013
4885 0000DCB8 08C0 <1> or al, al
4886 <1> ; tstb u.uid / super user?
4887 0000DCBA 7417 <1> jz short isown1
4888 <1> ; beq lf / yes, branch
4889 0000DCBC 3A05[494B0100] <1> cmp al, [i.uid]
4890 <1> ; cmpb i.uid,u.uid / no, is this the owner of
4891 <1> ; / the file
4892 <1> ;jne error
4893 <1> ; beq lf / yes
4894 <1> ; jmp error3 / no, error
4895 0000DCC2 740F <1> je short isown1
4896 <1>
4897 0000DCC4 C705[104F0100]0B00- <1> mov dword [u.error], ERR_NOT_OWNER ; 11
4897 0000DCCC 0000 <1>
4898 <1> ; 'permission denied !' error
4899 0000DCCE E91CE7FFFF <1> jmp error
4900 <1> isown1: ; 1:
4901 0000DCD3 E8EA140000 <1> call setimod
4902 <1> ; jsr r0,setimod / indicates
4903 <1> ; / i-node has been modified
4904 0000DCD8 58 <1> pop eax ; 2nd argument
4905 <1> ; mov (sp)+,r2 / mode is put in r2
4906 <1> ; / (u.off put on stack with 2nd arg)
4907 0000DCD9 C3 <1> retn
4908 <1> ; rts r0

```



```

4909 <1>
4910 <1> ;;arg: ; < get system call arguments >
4911 <1> ; 'arg' extracts an argument for a routine whose call is
4912 <1> ; of form:
4913 <1> ; sys 'routine' ; arg1
4914 <1> ; or
4915 <1> ; sys 'routine' ; arg1 ; arg2
4916 <1> ; or
4917 <1> ; sys 'routine' ; arg1;...;arg10 (sys exec)
4918 <1> ;
4919 <1> ; INPUTS ->
4920 <1> ; u.sp+18 - contains a pointer to one of arg1..argn
4921 <1> ; This pointers's value is actually the value of
4922 <1> ; update pc at the the trap to sysent (unkni) is
4923 <1> ; made to process the sys instruction
4924 <1> ; r0 - contains the return address for the routine
4925 <1> ; that called arg. The data in the word pointer
4926 <1> ; to by the return address is used as address
4927 <1> ; in which the extracted argument is stored
4928 <1> ;
4929 <1> ; OUTPUTS ->
4930 <1> ; 'address' - contains the extracted argument
4931 <1> ; u.sp+18 - is incremented by 2
4932 <1> ; r1 - contains the extracted argument
4933 <1> ; r0 - points to the next instruction to be
4934 <1> ; executed in the calling routine.
4935 <1> ;
4936 <1>
4937 <1> ; mov u.sp,r1
4938 <1> ; mov *18.(r1),*(r0)+ / put argument of system call
4939 <1> ; / into argument of arg2
4940 <1> ; add $2,18.(r1) / point pc on stack
4941 <1> ; / to next system argument
4942 <1> ; rts r0
4943 <1>
4944 <1> ;;arg2: ; < get system calls arguments - with file name pointer>
4945 <1> ; 'arg2' takes first argument in system call
4946 <1> ; (pointer to name of the file) and puts it in location
4947 <1> ; u.namep; takes second argument and puts it in u.off
4948 <1> ; and on top of the stack
4949 <1> ;
4950 <1> ; INPUTS ->
4951 <1> ; u.sp, r0
4952 <1> ;
4953 <1> ; OUTPUTS ->
4954 <1> ; u.namep
4955 <1> ; u.off
4956 <1> ; u.off pushed on stack
4957 <1> ; r1
4958 <1> ;
4959 <1>
4960 <1> ; jsr r0,arg; u.namep / u.namep contains value of
4961 <1> ; / first arg in sys call
4962 <1> ; jsr r0,arg; u.off / u.off contains value of
4963 <1> ; / second arg in sys call
4964 <1> ; mov r0,r1 / r0 points to calling routine
4965 <1> ; mov (sp),r0 / put operation code back in r0
4966 <1> ; mov u.off,(sp) / put pointer to second argument
4967 <1> ; / on stack
4968 <1> ; jmp (r1) / return to calling routine
4969 <1>
4970 <1> syschown: ; < change owner of file >
4971 <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
4972 <1> ; 20/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
4973 <1> ;
4974 <1> ; 'syschown' changes the owner of the file whose name is given
4975 <1> ; as null terminated string pointed to by 'name' has it's owner
4976 <1> ; changed to 'owner'
4977 <1> ;
4978 <1> ; Calling sequence:
4979 <1> ; syschown; name; owner
4980 <1> ; Arguments:
4981 <1> ; name - address of the file name
4982 <1> ; terminated by null byte.
4983 <1> ; owner - (new) owner (number/ID)
4984 <1> ;
4985 <1> ; Inputs: -
4986 <1> ; Outputs: -
4987 <1> ; .....
4988 <1> ;
4989 <1> ; Retro UNIX 8086 v1 modification:
4990 <1> ; 'syschown' system call has two arguments; so,
4991 <1> ; * 1st argument, name is pointed to by BX register
4992 <1> ; * 2nd argument, owner number is in CX register
4993 <1> ;
4994 <1> ; / name; owner
4995 0000DCDA E8B2FFFFFF <1> call isown
4996 <1> ; jsr r0,isown / get the i-node and check user status
4997 0000DCDF 803D[F84E0100]00 <1> cmp byte [u.uid], 0 ; 02/08/2013
4998 <1> ; tstb u.uid / super user
4999 0000DCE6 7418 <1> jz short syschown1
5000 <1> ; beq 2f / yes, 2f
5001 0000DCE8 F605[464B0100]20 <1> test byte [i.flgs], 20h ; 32
5002 <1> ; bit $40,i.flgs / no, set userid on execution?
5003 <1> ;jnz error
5004 <1> ; bne 3f / yes error, could create Trojan Horses
5005 0000DCEF 740F <1> jz short syschown1
5006 <1> ; 'permission denied !'
5007 0000DCF1 C705[104F0100]0B00- <1> mov dword [u.error], ERR_FILE_ACCESS ; 11
5008 0000DCF9 0000 <1>
5009 0000DCFB E9EFE6FFFF <1> jmp error
5009 <1> syschown1: ; 2:

```

```

5010      <1>      ; AL = owner (number/ID)
5011 0000DD00 A2[494B0100] <1>      mov     [i.uid], al ; 23/06/2015
5012      <1>      ; movb     r2,i.uid / no, put the new owners id
5013      <1>      ; / in the i-node
5014 0000DD05 E905E7FFFF <1>      jmp     sysret
5015      <1>      ; 1:
5016      <1>      ; jmp sysret4
5017      <1>      ; 3:
5018      <1>      ; jmp error
5019      <1>
5020      <1> sysstime: ; / get time of year
5021      <1>      ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
5022      <1>      ; 20/06/2013 (Retro UNIX 8086 v1)
5023      <1>      ;
5024      <1>      ; 20/06/2013
5025      <1>      ; 'sysstime' gets the time of the year.
5026      <1>      ; The present time is put on the stack.
5027      <1>      ;
5028      <1>      ; Calling sequence:
5029      <1>      ; sysstime
5030      <1>      ; Arguments: -
5031      <1>      ;
5032      <1>      ; Inputs: -
5033      <1>      ; Outputs: sp+2, sp+4 - present time
5034      <1>      ; .....
5035      <1>      ;
5036      <1>      ; Retro UNIX 8086 v1 modification:
5037      <1>      ; 'sysstime' system call will return to the user
5038      <1>      ; with unix time (epoch) in DX:AX register pair
5039      <1>      ;
5040      <1>      ; !! Major modification on original Unix v1 'sysstime'
5041      <1>      ; system call for PC compatibility !!
5042      <1>
5043 0000DD0A E8B6140000 <1>      call    epoch
5044 0000DD0F A3[AC4E0100] <1>      mov     [u.r0], eax
5045      <1>      ; mov s.time,4(sp)
5046      <1>      ; mov s.time+2,2(sp) / put the present time
5047      <1>      ; / on the stack
5048      <1>      ; br sysret4
5049 0000DD14 E9F6E6FFFF <1>      jmp     sysret
5050      <1>
5051      <1> sysstime: ; / set time
5052      <1>      ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
5053      <1>      ; 20/06/2013 - 02/08/2013 (Retro UNIX 8086 v1)
5054      <1>      ;
5055      <1>      ; 'sysstime' sets the time. Only super user can use this call.
5056      <1>      ;
5057      <1>      ; Calling sequence:
5058      <1>      ; sysstime
5059      <1>      ; Arguments: -
5060      <1>      ;
5061      <1>      ; Inputs: sp+2, sp+4 - time system is to be set to.
5062      <1>      ; Outputs: -
5063      <1>      ; .....
5064      <1>      ;
5065      <1>      ; Retro UNIX 8086 v1 modification:
5066      <1>      ; the user calls 'sysstime' with unix (epoch) time
5067      <1>      ; (to be set) is in CX:BX register pair as two arguments.
5068      <1>      ;
5069      <1>      ; Retro UNIX 8086 v1 argument transfer method 2 is used
5070      <1>      ; to get sysstime system call arguments from the user;
5071      <1>      ; * 1st argument, lowword of unix time is in BX register
5072      <1>      ; * 2nd argument, highword of unix time is in CX register
5073      <1>      ;
5074      <1>      ; !! Major modification on original Unix v1 'sysstime'
5075      <1>      ; system call for PC compatibility !!
5076      <1>
5077 0000DD19 803D[F84E0100]00 <1>      cmp     byte [u.uid], 0
5078      <1>      ; tstb u.uid / is user the super user
5079      <1>      ; ja error
5080      <1>      ; bne error4 / no, error
5081 0000DD20 760F <1>      jna     short systime1
5082      <1>      ; 'permission denied !'
5083 0000DD22 C705[104F0100]0B00- <1>      mov     dword [u.error], ERR_NOT_SUPERUSER ; 11
5084 0000DD2A 0000 <1>
5085 0000DD2C E9BEE6FFFF <1>      jmp     error
5086      <1> systime1:
5087      <1>      ; 23/06/2015 (Retro UNIX 386 v1 - 32 bit version)
5088      <1>      ; EBX = unix (epoch) time (from user)
5089      <1>      mov     eax, ebx
5090      <1>      call    set_date_time
5091      <1>      ; mov 4(sp),s.time
5092      <1>      ; mov 2(sp),s.time+2 / set the system time
5093      <1>      jmp     sysret
5094      <1>      ; br sysret4
5095      <1> sysbreak:
5096      <1>      ; 18/10/2015
5097      <1>      ; 07/10/2015
5098      <1>      ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
5099      <1>      ; 20/06/2013 - 24/03/2014 (Retro UNIX 8086 v1)
5100      <1>      ;
5101      <1>      ; 'sysbreak' sets the programs break points.
5102      <1>      ; It checks the current break point (u.break) to see if it is
5103      <1>      ; between "core" and the stack (sp). If it is, it is made an
5104      <1>      ; even address (if it was odd) and the area between u.break
5105      <1>      ; and the stack is cleared. The new breakpoint is then put
5106      <1>      ; in u.break and control is passed to 'sysret'.
5107      <1>      ;
5108      <1>      ; Calling sequence:
5109      <1>      ; sysbreak; addr
5110      <1>      ; Arguments: -

```

```

5111 <1> ;
5112 <1> ; Inputs: u.break - current breakpoint
5113 <1> ; Outputs: u.break - new breakpoint
5114 <1> ; area between old u.break and the stack (sp) is cleared.
5115 <1> ; .....
5116 <1> ;
5117 <1> ; Retro UNIX 8086 v1 modification:
5118 <1> ; The user/application program puts breakpoint address
5119 <1> ; in BX register as 'sysbreak' system call argument.
5120 <1> ; (argument transfer method 1)
5121 <1> ;
5122 <1> ; NOTE: Beginning of core is 0 in Retro UNIX 8086 v1 !
5123 <1> ; (('sysbreak' is not needed in Retro UNIX 8086 v1!))
5124 <1> ; NOTE:
5125 <1> ; 'sysbreak' clears extended part (beyond of previous
5126 <1> ; 'u.break' address) of user's memory for original unix's
5127 <1> ; 'bss' compatibility with Retro UNIX 8086 v1 (19/11/2013)
5128 <1>
5129 <1> ; mov u.break,r1 / move users break point to r1
5130 <1> ; cmp r1,$core / is it the same or lower than core?
5131 <1> ; blos lf / yes, lf
5132 <1> ; 23/06/2015
5133 0000DD3D 8B2D[D84E0100] <1> mov ebp, [u.break] ; virtual address (offset)
5134 <1> ;and ebp, ebp
5135 <1> ;jz short sysbreak_3
5136 <1> ; Retro UNIX 386 v1 NOTE: u.break points to virtual address !!!
5137 <1> ; (Even break point address is not needed for Retro UNIX 386 v1)
5138 0000DD43 8B15[A44E0100] <1> mov edx, [u.sp] ; kernel stack at the beginning of sys call
5139 0000DD49 83C20C <1> add edx, 12 ; EIP -4-> CS -4-> EFLAGS -4-> ESP (user)
5140 <1> ; 07/10/2015
5141 0000DD4C 891D[D84E0100] <1> mov [u.break], ebx ; virtual address !!!
5142 <1> ;
5143 0000DD52 3B1A <1> cmp ebx, [edx] ; compare new break point with
5144 <1> ; with top of user's stack (virtual!)
5145 0000DD54 7327 <1> jnb short sysbreak_3
5146 <1> ; cmp r1,sp / is it the same or higher
5147 <1> ; / than the stack?
5148 <1> ; bhis lf / yes, lf
5149 0000DD56 89DE <1> mov esi, ebx
5150 0000DD58 29EE <1> sub esi, ebp ; new break point - old break point
5151 0000DD5A 7621 <1> jna short sysbreak_3
5152 <1> ;push ebx
5153 <1> sysbreak_1:
5154 0000DD5C 89EB <1> mov ebx, ebp
5155 0000DD5E E82B75FFFF <1> call get_physical_addr ; get physical address
5156 0000DD63 0F82ADFEFFFF <1> jc tr_addr_nm_err
5157 <1> ; 18/10/2015
5158 0000DD69 89C7 <1> mov edi, eax
5159 0000DD6B 29C0 <1> sub eax, eax ; 0
5160 <1> ; ECX = remain byte count in page (1-4096)
5161 0000DD6D 39CE <1> cmp esi, ecx
5162 0000DD6F 7302 <1> jnb short sysbreak_2
5163 0000DD71 89F1 <1> mov ecx, esi
5164 <1> sysbreak_2:
5165 0000DD73 29CE <1> sub esi, ecx
5166 0000DD75 01CD <1> add ebp, ecx
5167 0000DD77 F3AA <1> rep stosb
5168 0000DD79 09F6 <1> or esi, esi
5169 0000DD7B 75DF <1> jnz short sysbreak_1
5170 <1> ;
5171 <1> ; bit $1,r1 / is it an odd address
5172 <1> ; beq 2f / no, its even
5173 <1> ; clrb (r1)+ / yes, make it even
5174 <1> ; 2: / clear area between the break point and the stack
5175 <1> ; cmp r1,sp / is it higher or same than the stack
5176 <1> ; bhis lf / yes, quit
5177 <1> ; clr (r1)+ / clear word
5178 <1> ; br 2b / go back
5179 <1> ;pop ebx
5180 <1> sysbreak_3: ; 1:
5181 <1> ;mov [u.break], ebx ; virtual address !!!
5182 <1> ; jsr r0,arg; u.break / put the "address"
5183 <1> ; / in u.break (set new break point)
5184 <1> ; br sysret4 / br sysret
5185 0000DD7D E98DE6FFFF <1> jmp sysret
5186 <1>
5187 <1> sysseek: ; / moves read write pointer in an fsp entry
5188 <1> ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)
5189 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5190 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
5191 <1> ;
5192 <1> ; 'sysseek' changes the r/w pointer of (3rd word of in an
5193 <1> ; fsp entry) of an open file whose file descriptor is in u.r0.
5194 <1> ; The file descriptor refers to a file open for reading or
5195 <1> ; writing. The read (or write) pointer is set as follows:
5196 <1> ; * if 'ptrname' is 0, the pointer is set to offset.
5197 <1> ; * if 'ptrname' is 1, the pointer is set to its
5198 <1> ; current location plus offset.
5199 <1> ; * if 'ptrname' is 2, the pointer is set to the
5200 <1> ; size of file plus offset.
5201 <1> ; The error bit (e-bit) is set for an undefined descriptor.
5202 <1> ;
5203 <1> ; Calling sequence:
5204 <1> ; sysseek; offset; ptrname
5205 <1> ; Arguments:
5206 <1> ; offset - number of bytes desired to move
5207 <1> ; the r/w pointer
5208 <1> ; ptrname - a switch indicated above
5209 <1> ;
5210 <1> ; Inputs: r0 - file descriptor
5211 <1> ; Outputs: -
5212 <1> ; .....

```

```

5213 <1> ;
5214 <1> ; Retro UNIX 8086 v1 modification:
5215 <1> ; 'sysseek' system call has three arguments; so,
5216 <1> ; * 1st argument, file descriptor is in BX (BL) register
5217 <1> ; * 2nd argument, offset is in CX register
5218 <1> ; * 3rd argument, ptrname/switch is in DX (DL) register
5219 <1>
5220 0000DD82 E821000000 <1> call seektell
5221 <1> ; EAX = Current R/W pointer of the file
5222 <1> ; EBX = [u.fofp]
5223 <1> ; [u.base] = offset (ECX input)
5224 <1>
5225 0000DD87 0305[CC4E0100] <1> add eax, [u.base]
5226 0000DD8D 8903 <1> mov [ebx], eax
5227 0000DD8F E97BE6FFFF <1> jmp sysret
5228 <1>
5229 <1> systell: ; / get the r/w pointer
5230 <1> ; 06/11/2016 - TRDOS 386 (TRDOS v2.0) - temporary !-
5231 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5232 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
5233 <1> ;
5234 <1> ; Retro UNIX 8086 v1 modification:
5235 <1> ; ! 'systell' does not work in original UNIX v1,
5236 <1> ; it returns with error !
5237 <1> ; Inputs: r0 - file descriptor
5238 <1> ; Outputs: r0 - file r/w pointer
5239 <1>
5240 <1> ;xor ecx, ecx ; 0
5241 0000DD94 BA01000000 <1> mov edx, 1 ; 05/08/2013
5242 <1> ;call seektell
5243 0000DD99 E810000000 <1> call seektell0 ; 05/08/2013
5244 <1> ; 06/11/2016
5245 <1> ; ; move eax, [ebx]
5246 0000DD9E A3[AC4E0100] <1> mov [u.r0], eax
5247 0000DDA3 E967E6FFFF <1> jmp sysret
5248 <1>
5249 <1> ; Original unix v1 'systell' system call:
5250 <1> ; jsr r0,seektell
5251 <1> ; br error4
5252 <1>
5253 <1> seektell:
5254 <1> ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)
5255 <1> ; 03/01/2016
5256 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5257 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
5258 <1> ;
5259 <1> ; 'seektell' puts the arguments from sysseek and systell
5260 <1> ; call in u.base and u.count. It then gets the i-number of
5261 <1> ; the file from the file descriptor in u.r0 and by calling
5262 <1> ; getf. The i-node is brought into core and then u.count
5263 <1> ; is checked to see it is a 0, 1, or 2.
5264 <1> ; If it is 0 - u.count stays the same
5265 <1> ; 1 - u.count = offset (u.fofp)
5266 <1> ; 2 - u.count = i.size (size of file)
5267 <1> ;
5268 <1> ; !! Retro UNIX 8086 v1 modification:
5269 <1> ; Argument 1, file descriptor is in BX;
5270 <1> ; Argument 2, offset is in CX;
5271 <1> ; Argument 3, ptrname/switch is in DX register.
5272 <1> ;
5273 <1> ; ((Return -> eax = base for offset (position= base+offset))
5274 <1> ;
5275 0000DDA8 890D[CC4E0100] <1> mov [u.base], ecx ; offset
5276 <1> seektell0:
5277 0000DDAE 8915[D04E0100] <1> mov [u.count], edx
5278 <1> ; EBX = file descriptor (file number)
5279 0000DDB4 E898FCFFFF <1> call getf1
5280 <1> ; EAX = First cluster of the file
5281 <1> ; EBX = File number (Open file number)
5282 <1> ; [u.fofp] = Pointer to File pointer
5283 <1> ; [i.size] = File size
5284 <1>
5285 0000DDB9 09C0 <1> or eax, eax
5286 0000DDBB 7514 <1> jnz short seektell1
5287 <1>
5288 0000DDBD B80A000000 <1> mov eax, ERR_FILE_NOT_OPEN
5289 0000DDC2 A3[AC4E0100] <1> mov [u.r0], eax
5290 0000DDC7 A3[104F0100] <1> mov dword [u.error], eax ; 'file not open !'
5291 0000DDCC E91EE6FFFF <1> jmp error
5292 <1>
5293 <1> seektell1:
5294 0000DDD1 8B1D[BC4E0100] <1> mov ebx, [u.fofp]
5295 0000DDD7 803D[D04E0100]01 <1> cmp byte [u.count], 1
5296 0000DDDE 7705 <1> ja short seektell2
5297 0000DDE0 7409 <1> je short seektell3
5298 0000DDE2 31C0 <1> xor eax, eax
5299 0000DDE4 C3 <1> retn
5300 <1>
5301 <1> seektell2:
5302 0000DDE5 A1[9D4F0100] <1> mov eax, [i.size]
5303 0000DDEA C3 <1> retn
5304 <1>
5305 <1> seektell3:
5306 0000DDEB 8B03 <1> mov eax, [ebx]
5307 0000DDED C3 <1> retn
5308 <1>
5309 <1> sysintr: ; / set interrupt handling
5310 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5311 <1> ; 07/07/2013 (Retro UNIX 8086 v1)
5312 <1> ;
5313 <1> ; 'sysintr' sets the interrupt handling value. It puts
5314 <1> ; argument of its call in u.intr then branches into 'sysquit'

```



```

5315 <1> ; routine. u.tty is checked if to see if a control tty exists.
5316 <1> ; If one does the interrupt character in the tty buffer is
5317 <1> ; cleared and 'sysret' is called. If one does not exits
5318 <1> ; 'sysret' is just called.
5319 <1> ;
5320 <1> ; Calling sequence:
5321 <1> ; sysintr; arg
5322 <1> ; Argument:
5323 <1> ; arg - if 0, interrupts (ASCII DELETE) are ignored.
5324 <1> ; - if 1, interrupts cause their normal result
5325 <1> ; i.e force an exit.
5326 <1> ; - if arg is a location within the program,
5327 <1> ; control is passed to that location when
5328 <1> ; an interrupt occurs.
5329 <1> ; Inputs: -
5330 <1> ; Outputs: -
5331 <1> ; .....
5332 <1> ;
5333 <1> ; Retro UNIX 8086 v1 modification:
5334 <1> ; 'sysintr' system call sets u.intr to value of BX
5335 <1> ; then branches into sysquit.
5336 <1> ;
5337 0000DDEE 66891D[F24E0100] <1> mov [u.intr], bx
5338 <1> ; jsr r0,arg; u.intr / put the argument in u.intr
5339 <1> ; br 1f / go into quit routine
5340 0000DDF5 E915E6FFFF <1> jmp sysret
5341 <1>
5342 <1> sysquit:
5343 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5344 <1> ; 07/07/2013 (Retro UNIX 8086 v1)
5345 <1> ;
5346 <1> ; 'sysquit' turns off the quit signal. it puts the argument of
5347 <1> ; the call in u.quit. u.tty is checked if to see if a control
5348 <1> ; tty exists. If one does the interrupt character in the tty
5349 <1> ; buffer is cleared and 'sysret' is called. If one does not exits
5350 <1> ; 'sysret' is just called.
5351 <1> ;
5352 <1> ; Calling sequence:
5353 <1> ; sysquit; arg
5354 <1> ; Argument:
5355 <1> ; arg - if 0, this call disables quit signals from the
5356 <1> ; typewriter (ASCII FS)
5357 <1> ; - if 1, quits are re-enabled and cause execution to
5358 <1> ; cease and a core image to be produced.
5359 <1> ; i.e force an exit.
5360 <1> ; - if arg is an address in the program,
5361 <1> ; a quit causes control to sent to that
5362 <1> ; location.
5363 <1> ; Inputs: -
5364 <1> ; Outputs: -
5365 <1> ; .....
5366 <1> ;
5367 <1> ; Retro UNIX 8086 v1 modification:
5368 <1> ; 'sysquit' system call sets u.quit to value of BX
5369 <1> ; then branches into 'sysret'.
5370 <1> ;
5371 0000DDFA 66891D[F44E0100] <1> mov [u.quit], bx
5372 0000DE01 E909E6FFFF <1> jmp sysret
5373 <1> ; jsr r0,arg; u.quit / put argument in u.quit
5374 <1> ;1:
5375 <1> ; mov u.ttyp,r1 / move pointer to control tty buffer
5376 <1> ; / to r1
5377 <1> ; beq sysret4 / return to user
5378 <1> ; clrb 6(r1) / clear the interrupt character
5379 <1> ; / in the tty buffer
5380 <1> ; br sysret4 / return to user
5381 <1>
5382 <1> syssetuid: ; / set process id
5383 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5384 <1> ; 07/07/2013 - 02/08/2013 (Retro UNIX 8086 v1)
5385 <1> ;
5386 <1> ; 'syssetuid' sets the user id (u.uid) of the current process
5387 <1> ; to the process id in (u.r0). Both the effective user and
5388 <1> ; u.uid and the real user u.ruid are set to this.
5389 <1> ; Only the super user can make this call.
5390 <1> ;
5391 <1> ; Calling sequence:
5392 <1> ; syssetuid
5393 <1> ; Arguments: -
5394 <1> ;
5395 <1> ; Inputs: (u.r0) - contains the process id.
5396 <1> ; Outputs: -
5397 <1> ; .....
5398 <1> ;
5399 <1> ; Retro UNIX 8086 v1 modification:
5400 <1> ; BL contains the (new) user ID of the current process
5401 <1>
5402 <1> ; movb *u.r0,r1 / move process id (number) to r1
5403 0000DE06 3A1D[F94E0100] <1> cmp bl, [u.ruid]
5404 <1> ; cmpb r1,u.ruid / is it equal to the real user
5405 <1> ; / id number
5406 0000DE0C 741E <1> je short setuid1
5407 <1> ; beq 1f / yes
5408 0000DE0E 803D[F84E0100]00 <1> cmp byte [u.uid], 0 ; 02/08/2013
5409 <1> ; tstb u.uid / no, is current user the super user?
5410 <1> ; ja error
5411 <1> ; bne error4 / no, error
5412 0000DE15 760F <1> jna short setuid0
5413 0000DE17 C705[104F0100]0B00- <1> mov dword [u.error], ERR_NOT_SUPERUSER ; 11
5414 0000DE1F 0000 <1>
5415 0000DE21 E9C9E5FFFF <1> jmp error

```

```

5416                                <1> setuid0:
5417 0000DE26 881D[F94E0100]        <1>     mov    [u.ruid], bl
5418                                <1> setuid1: ; 1:
5419 0000DE2C 881D[F84E0100]        <1>     mov    [u.uid], bl ; 02/08/2013
5420                                <1>           ; movb r1,u.uid / put process id in u.uid
5421                                <1>           ; movb r1,u.ruid / put process id in u.ruid
5422 0000DE32 E9D8E5FFFF            <1>     jmp     sysret
5423                                <1>           ; br sysret4 / system return
5424                                <1>
5425                                <1> sysgetuid: ; < get user id >
5426                                <1>           ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5427                                <1>           ; 07/07/2013 (Retro UNIX 8086 v1)
5428                                <1>           ;
5429                                <1>           ; 'sysgetuid' returns the real user ID of the current process.
5430                                <1>           ; The real user ID identifies the person who is logged in,
5431                                <1>           ; in contradistinction to the effective user ID, which
5432                                <1>           ; determines his access permission at each moment. It is thus
5433                                <1>           ; useful to programs which operate using the 'set user ID'
5434                                <1>           ; mode, to find out who invoked them.
5435                                <1>           ;
5436                                <1>           ; Calling sequence:
5437                                <1>           ;     syssetuid
5438                                <1>           ; Arguments: -
5439                                <1>           ;
5440                                <1>           ; Inputs: -
5441                                <1>           ; Outputs: (u.r0) - contains the real user's id.
5442                                <1>           ; .....
5443                                <1>           ;
5444                                <1>           ; Retro UNIX 8086 v1 modification:
5445                                <1>           ;     AL contains the real user ID at return.
5446                                <1>           ;
5447 0000DE37 0FB605[F94E0100]        <1>     movzx  eax, byte [u.ruid]
5448 0000DE3E A3[AC4E0100]            <1>     mov     [u.r0], eax
5449                                <1>           ; movb u.ruid,*u.r0 / move the real user id to (u.r0)
5450 0000DE43 E9C7E5FFFF            <1>     jmp     sysret
5451                                <1>           ; br sysret4 / system return, sysret
5452                                <1>
5453                                <1> anyi:
5454                                <1>           ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
5455                                <1>           ; Major Modification!
5456                                <1>           ; TRDOS 386 does not permit to delete a file while it is open
5457                                <1>           ; The role of 'anyi' procedure has been changed to ensure that.
5458                                <1>           ;
5459                                <1>           ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
5460                                <1>           ; 25/04/2013 (Retro UNIX 8086 v1)
5461                                <1>           ;
5462                                <1>           ; 'anyi' is called if a file deleted while open.
5463                                <1>           ; "anyi" checks to see if someone else has opened this file.
5464                                <1>           ;
5465                                <1>           ; INPUTS ->
5466                                <1>           ;     r1 - contains an i-number
5467                                <1>           ;     fsp - start of table containing open files
5468                                <1>           ;
5469                                <1>           ; OUTPUTS ->
5470                                <1>           ;     "deleted" flag set in fsp entry of another occurrence of
5471                                <1>           ;     this file and r2 points 1st word of this fsp entry.
5472                                <1>           ;     if file not found - bit in i-node map is cleared
5473                                <1>           ;     (i-node is freed)
5474                                <1>           ;     all blocks related to i-node are freed
5475                                <1>           ;     all flags in i-node are cleared
5476                                <1>           ; ((AX = R1)) input
5477                                <1>           ;
5478                                <1>           ; (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
5479                                <1>           ; ((Modified registers: EDI, ECX, EBX, ESI, EDI, EBP))
5480                                <1>           ;
5481                                <1>           ; / r1 contains an i-number
5482                                <1>
5483                                <1>           ; TRDOS 386 (06/10/2016)
5484                                <1>           ;
5485                                <1>           ; INPUT:
5486                                <1>           ;     EAX = First Cluster
5487                                <1>           ;     DL = Logical DOS Drive Number
5488                                <1>           ;
5489                                <1>           ; OUTPUT:
5490                                <1>           ;     CF = 1 -> EBX = File Handle/Number/Index
5491                                <1>           ;     CF = 0 -> EBX = 0
5492                                <1>           ;
5493                                <1>           ; Modified Registers: EBX
5494                                <1>
5495 0000DE48 31DB                    <1>     xor     ebx, ebx
5496                                <1> anyi_0:
5497 0000DE4A 80BB[4A490100]00        <1>     cmp     byte [ebx+OF_MODE], 0 ; 0 = empty entry
5498 0000DE51 770A                    <1>     ja      short anyi_2 ; 1 (r), 2 (w) or 3 (r&w)
5499                                <1> anyi_1:
5500 0000DE53 FEC3                    <1>     inc     bl
5501 0000DE55 80FB0A                  <1>     cmp     bl, OPENFILES ; max. count of open files
5502 0000DE58 72F0                    <1>     jnb     short anyi_0
5503 0000DE5A 31C0                    <1>     xor     eax, eax
5504 0000DE5C C3                      <1>     retn
5505                                <1> anyi_2:
5506 0000DE5D 3A93[40490100]          <1>     cmp     dl, [ebx+OF_DRIVE]
5507 0000DE63 75EE                    <1>     jne     short anyi_1
5508 0000DE65 66C1E302                <1>     shl     bx, 2 ; *4 (dword offset)
5509 0000DE69 3B83[18490100]          <1>     cmp     eax, [ebx+OF_FCLUSTER]
5510 0000DE6F 7406                    <1>     je      short anyi_3
5511 0000DE71 66C1EB02                <1>     shr     bx, 2 ; /4 (byte offset)
5512 0000DE75 EBDC                    <1>     jmp     short anyi_1
5513                                <1> anyi_3:
5514 0000DE77 66C1EB02                <1>     shr     bx, 2 ; /4 (bytes offset) (index)
5515 0000DE7B F9                      <1>     stc
5516 0000DE7C C3                      <1>     retn
5517                                <1>

```

```

5518 <1> ; Retro UNIX 386 v1 Kernel (v0.2) - u7.s
5519 <1> ; Last Modification: 14/11/2015
5520 <1>
5521 <1> sysmount: ; / mount file system
5522 <1> ; 24/10/2016 - TRDOS 386 (TRDOS v2.0)
5523 <1> ; temporary !
5524 0000DE7D B80F000000 <1> mov eax, ERR_DRV_NOT_RDY ; drive not ready !
5525 0000DE82 A3[104F0100] <1> mov [u.error], eax
5526 0000DE87 A3[AC4E0100] <1> mov [u.r0], eax
5527 0000DE8C E95EE5FFFF <1> jmp error
5528 <1>
5529 <1> sysumount: ; / special dismount file system
5530 <1> ; 24/10/2016 - TRDOS 386 (TRDOS v2.0)
5531 <1> ; temporary !
5532 0000DE91 B80F000000 <1> mov eax, ERR_DRV_NOT_RDY ; drive not ready !
5533 0000DE96 A3[104F0100] <1> mov [u.error], eax
5534 0000DE9B A3[AC4E0100] <1> mov [u.r0], eax
5535 0000DEA0 E94AE5FFFF <1> jmp error
5536 <1>
5537 <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS9.INC
5538 <1> ; Last Modification: 09/12/2015
5539 <1>
5540 <1> sysssleep:
5541 <1> ; 29/06/2015 - (Retro UNIX 386 v1)
5542 <1> ; 11/06/2014 - (Retro UNIX 8086 v1)
5543 <1> ;
5544 <1> ; Retro UNIX 8086 v1 feature only
5545 <1> ; (INPUT -> none)
5546 <1> ;
5547 0000DEA5 0FB61D[FB4E0100] <1> movzx ebx, byte [u.uno] ; process number
5548 0000DEAC 8AA3[C54B0100] <1> mov ah, [ebx+p.ttyc-1] ; current/console tty
5549 0000DEB2 E80F130000 <1> call sleep
5550 0000DEB7 E953E5FFFF <1> jmp sysret
5551 <1>
5552 <1> _vp_clr:
5553 <1> ; Reset/Clear Video Page
5554 <1> ;
5555 <1> ; 30/06/2015 - (Retro UNIX 386 v1)
5556 <1> ; 21/05/2013 - 30/10/2013(Retro UNIX 8086 v1) (U0.ASM)
5557 <1> ;
5558 <1> ; Retro UNIX 8086 v1 feature only !
5559 <1> ;
5560 <1> ; INPUTS ->
5561 <1> ; BH = video page number
5562 <1> ;
5563 <1> ; OUTPUT ->
5564 <1> ; none
5565 <1> ; ((Modified registers: eAX, BH, eCX, eDX, eSI, eDI))
5566 <1> ;
5567 <1> ; 04/12/2013
5568 0000DEBC 28C0 <1> sub al, al
5569 <1> ; al = 0 (clear video page)
5570 <1> ; bh = video page ; 13/05/2016
5571 0000DEBE B407 <1> mov ah, 07h
5572 <1> ; ah = 7 (attribute/color)
5573 0000DEC0 6631C9 <1> xor cx, cx ; 0, left upper column (cl) & row (cl)
5574 0000DEC3 66BA4F18 <1> mov dx, 184Fh ; right lower column & row (dl=24, dh=79)
5575 0000DEC7 E83E3BFFFF <1> call _scroll_up
5576 <1> ; bh = video page
5577 0000DECC 6631D2 <1> xor dx, dx ; 0 (cursor position)
5578 0000DECF E9743EFFFF <1> jmp _set_cpos
5579 <1>
5580 <1> sysmsg:
5581 <1> ; 13/05/2016
5582 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
5583 <1> ; 01/07/2015 - 11/11/2015 (Retro UNIX 386 v1)
5584 <1> ; Print user-application message on user's console tty
5585 <1> ;
5586 <1> ; Input -> EBX = Message address
5587 <1> ; ECX = Message length (max. 255)
5588 <1> ; DL = Color (IBM PC Rombios color attributes)
5589 <1> ;
5590 0000DED4 81F9FF000000 <1> cmp ecx, MAX_MSG_LEN ; 255
5591 0000DEDA 0F872FE5FFFF <1> ja sysret ; nothing to do with big message size
5592 0000DEE0 08C9 <1> or cl, cl
5593 0000DEE2 0F8427E5FFFF <1> jz sysret
5594 0000DEE8 20D2 <1> and dl, dl
5595 0000DEEA 7502 <1> jnz short sysmsg0
5596 0000DEEC B207 <1> mov dl, 07h ; default color
5597 <1> ; (black background, light gray character)
5598 <1> sysmsg0:
5599 0000DEEE 891D[CC4E0100] <1> mov [u.base], ebx
5600 0000DEF4 8815[4F380100] <1> mov [ccolor], dl ; color attributes
5601 0000DEFA 89E5 <1> mov ebp, esp
5602 0000DEFC 31DB <1> xor ebx, ebx ; 0
5603 0000DEFE 891D[D44E0100] <1> mov [u.nread], ebx ; 0
5604 <1> ;
5605 0000DF04 381D[0E4F0100] <1> cmp [u.kcall], bl ; 0
5606 0000DF0A 7769 <1> ja short sysmsgk ; Temporary (01/07/2015)
5607 <1> ;
5608 0000DF0C 890D[D04E0100] <1> mov [u.count], ecx
5609 0000DF12 41 <1> inc ecx ; + 00h ; ASCIIIZ
5610 0000DF13 29CC <1> sub esp, ecx
5611 0000DF15 89E7 <1> mov edi, esp
5612 0000DF17 89E6 <1> mov esi, esp
5613 0000DF19 66891D[0C4F0100] <1> mov [u.pcount], bx ; reset page (phy. addr.) counter
5614 <1> ; 11/11/2015
5615 0000DF20 8A25[DC4E0100] <1> mov ah, [u.ttyp] ; recent open tty
5616 <1> ; 0 = none
5617 0000DF26 FECC <1> dec ah
5618 0000DF28 790C <1> jns short sysmsg1
5619 0000DF2A 8A1D[FB4E0100] <1> mov bl, [u.uno] ; process number

```

403

Address	Disassembly	Comment
5620	0000DF30 8AA3[C54B0100]	<1> mov ah, [ebx+p.ttyc-1] ; user's (process's) console tty
5621		<1> sysmsg1:
5622	0000DF36 8825[DE4E0100]	<1> mov [u.ttyn], ah
5623		<1> sysmsg2:
5624	0000DF3C E82F080000	<1> call cpass
5625	0000DF41 7416	<1> jz short sysmsg5
5626	0000DF43 AA	<1> stosb
5627	0000DF44 20C0	<1> and al, al
5628	0000DF46 75F4	<1> jnz short sysmsg2
5629		<1> sysmsg3:
5630	0000DF48 80FC07	<1> cmp ah, 7 ; tty number
5631	0000DF4B 7711	<1> ja short sysmsg6 ; serial port
5632	0000DF4D E83E000000	<1> call print_cmsg
5633		<1> sysmsg4:
5634	0000DF52 89EC	<1> mov esp, ebp
5635	0000DF54 E9B6E4FFFF	<1> jmp sysret
5636		<1> sysmsg5:
5637	0000DF59 C60700	<1> mov byte [edi], 0
5638	0000DF5C EBEA	<1> jmp short sysmsg3
5639		<1> sysmsg6:
5640	0000DF5E 8A06	<1> mov al, [esi]
5641	0000DF60 E85E120000	<1> call sndc
5642	0000DF65 72EB	<1> jc short sysmsg4
5643	0000DF67 803E00	<1> cmp byte [esi], 0 ; 0 is stop character
5644	0000DF6A 76E6	<1> jna short sysmsg4
5645	0000DF6C 46	<1> inc esi
5646	0000DF6D 8A25[DE4E0100]	<1> mov ah, [u.ttyn]
5647	0000DF73 EBE9	<1> jmp short sysmsg6
5648		<1>
5649		<1> sysmsgk: ; Temporary (01/07/2015)
5650		<1> ; The message has been sent by Kernel (ASCIIZ string)
5651		<1> ; (ECX -character count- will not be considered)
5652	0000DF75 8B35[CC4E0100]	<1> mov esi, [u.base]
5653	0000DF7B 8A25[4E380100]	<1> mov ah, [ptty] ; present/current screen (video page)
5654	0000DF81 8825[DE4E0100]	<1> mov [u.ttyn], ah
5655	0000DF87 C605[0E4F0100]00	<1> mov byte [u.kcall], 0
5656	0000DF8E EBB8	<1> jmp short sysmsg3
5657		<1>
5658		<1> print_cmsg:
5659		<1> ; 13/05/2016 - TRDOS 386 (TRDOS v2.0)
5660		<1> ; 01/07/2015 (Retro UNIX 386 v1)
5661		<1> ;
5662		<1> ; print message (on user's console tty)
5663		<1> ; with requested color
5664		<1> ;
5665		<1> ; INPUTS:
5666		<1> ; esi = message address
5667		<1> ; [u.ttyn] = tty number (0 to 7)
5668		<1> ; [ccolor] = color attributes (IBM PC BIOS colors)
5669		<1>
5670	0000DF90 8A3D[DE4E0100]	<1> mov bh, [u.ttyn]
5671		<1> ;mov bh, ah
5672		<1>
5673	0000DF96 AC	<1> lodsb
5674		<1> pcmsg1:
5675	0000DF97 56	<1> push esi
5676	0000DF98 8A1D[4F380100]	<1> mov bl, [ccolor]
5677		<1> ;mov bh, [u.ttyn]
5678	0000DF9E E80F3DFFFF	<1> call _write_tty
5679	0000DFA3 5E	<1> pop esi
5680	0000DFA4 AC	<1> lodsb
5681	0000DFA5 20C0	<1> and al, al ; 0
5682	0000DFA7 75EE	<1> jnz short pcmsg1
5683	0000DFA9 C3	<1> retn
5684		<1>
5685		<1> sysgeterr:
5686		<1> ; 09/12/2015
5687		<1> ; 21/09/2015 - (Retro UNIX 386 v1 feature only!)
5688		<1> ; Get last error number or page fault count
5689		<1> ; (for debugging)
5690		<1> ;
5691		<1> ; Input -> EBX = return type
5692		<1> ; 0 = last error code (which is in 'u.error')
5693		<1> ; FFFFFFFFh = page fault count for running process
5694		<1> ; FFFFFFFEh = total page fault count
5695		<1> ; 1 .. FFFFFFFDh = undefined
5696		<1> ;
5697		<1> ; Output -> EAX = last error number or page fault count
5698		<1> ; (depending on EBX input)
5699		<1> ;
5700	0000DFAA 21DB	<1> and ebx, ebx
5701	0000DFAC 750B	<1> jnz short glerr_2
5702		<1> glerr_0:
5703	0000DFAE A1[104F0100]	<1> mov eax, [u.error]
5704		<1> glerr_1:
5705	0000DFB3 A3[AC4E0100]	<1> mov [u.r0], eax
5706	0000DFB8 C3	<1> retn


```

5722 <1> ; 24/10/2016
5723 <1> ; 24/04/2016, 02/05/2016, 03/05/2016, 06/05/2016
5724 <1> ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)
5725 <1> ; 23/10/2015 (Retro UNIX 386 v1, 'sysexec')
5726 <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
5727 <1> ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
5728 <1> ; EAX = First Cluster number
5729 <1> ; EDX = File Size
5730 <1> ; ESI = Argument list address
5731 <1> ; [argc] = argument count
5732 <1> ; [u.nread] = argument list length
5733 <1> ; [esp] = return address to the caller (*)
5734 <1> ;
5735 0000DFCD 8935[944F0100] <1> mov [argv], esi
5736 0000DFD3 8915[9D4F0100] <1> mov [i.size], edx
5737 0000DFD9 A3[994F0100] <1> mov [ii], eax
5738 <1>
5739 <1> ;sti ; 07/01/2017
5740 <1> ;mov eax, [k_page_dir]
5741 <1> ;mov [u.pgdir], eax
5742 0000DFDE 31C0 <1> xor eax, eax ; clc ; *** ; 04/01/2017
5743 <1> ;mov [u.r0], eax ; 0 ; 07/01/2017
5744 <1>
5745 <1> ; 06/05/2016
5746 <1> ; Set 'sysexit' return order to MainProg
5747 <1> ;
5748 0000DFE0 58 <1> pop eax ; * 'loc_load_and_run_file_8:' address
5749 <1> ; ; 22/01/2017
5750 <1> ; ;cli ; 07/01/2017
5751 0000DFE1 8B25[BC370100] <1> mov esp, [tss.esp0]
5752 <1> ;
5753 <1> ; 'loc_load_run_file_8' address has
5754 <1> ; 'jmp loc_file_rw_restore_retn' instruction
5755 <1> ; 'loc_file_rw_restore_retn:' will return to
5756 <1> ; [mainprog_return_addr]
5757 <1> ; just after 'call command_interpreter'
5758 <1> ;
5759 0000DFE7 68[9B620000] <1> push _end_of_mainprog ; we must not return to here !
5760 0000DFEC FF35[A4450100] <1> push dword [mainprog_return_addr]
5761 0000DFF2 89E5 <1> mov ebp, esp ; **
5762 <1> ;
5763 0000DFF4 9C <1> pushfd ; EFLAGS ; IRETD ; ***
5764 0000DFF5 6A08 <1> push KCODE ; cs ; IRETD
5765 0000DFF7 50 <1> push eax ; * (eip) ; IRETD
5766 0000DFF8 8925[A44E0100] <1> mov [u.sp], esp
5767 <1> ;mov byte [u.quant], time_count
5768 0000DFFE 1E <1> push ds
5769 0000DFFF 06 <1> push es
5770 0000E000 0FA0 <1> push fs
5771 0000E002 0FA8 <1> push gs
5772 <1> ;mov eax, [u.r0]
5773 0000E004 29C0 <1> sub eax, eax
5774 0000E006 60 <1> pushad
5775 0000E007 68[0FC40000] <1> push sysret
5776 <1> ;push sysrell ; 07/01/2017
5777 0000E00C 8925[A84E0100] <1> mov [u.usp], esp
5778 <1> ;
5779 0000E012 E85E060000 <1> call wswap ; Save MainProg (process 1) 'u' structure
5780 <1> ; and registers for return (from program)
5781 0000E017 89EC <1> mov esp, ebp ; **
5782 <1> ; ;22/01/2017
5783 <1> ; ;sti ; 07/01/2017
5784 0000E019 50 <1> push eax ; * 'loc_load_and_run_file_8:' address
5785 <1> ;
5786 <1> ; ; 02/05/2016
5787 <1> ; ; Create a new process (parent: MainProg)
5788 0000E01A 31F6 <1> xor esi, esi
5789 <1> cnpm_1: ; search p.stat table for unused process number
5790 0000E01C 46 <1> inc esi
5791 0000E01D 80BE[F54B0100]00 <1> cmp byte [esi+p.stat-1], 0 ; SFREE
5792 <1> ; is process active, unused, dead
5793 0000E024 760B <1> jna short cnpm_2 ; it's unused so branch
5794 0000E026 6683FE10 <1> cmp si, nproc ; all processes checked
5795 0000E02A 72F0 <1> jb short cnpm_1 ; no, branch back
5796 0000E02C E9E082FFFF <1> jmp panic
5797 <1> cnpm_2:
5798 0000E031 A1[004F0100] <1> mov eax, [u.pgdir] ; page directory of MainProg
5799 0000E036 A3[044F0100] <1> mov [u.ppgdir], eax ; parent's page directory
5800 0000E03B E8396BFFFF <1> call allocate_page
5801 0000E040 0F82CB82FFFF <1> jc panic
5802 <1> ; EAX = UPAGE (user structure page) address
5803 0000E046 A3[FC4E0100] <1> mov [u.upage], eax ; memory page for 'user' struct (child)
5804 0000E04B 89F7 <1> mov edi, esi
5805 0000E04D 66C1E702 <1> shl di, 2
5806 0000E051 8987[024C0100] <1> mov [edi+p.upage-4], eax ; memory page for 'user' struct
5807 0000E057 E8976BFFFF <1> call clear_page ; 03/05/2016
5808 <1> ;movzx eax, byte [p.ttyc] ; console tty (for MainProg)
5809 0000E05C 6629C0 <1> sub ax, ax ; 0
5810 0000E05F 668986[C54B0100] <1> mov [esi+p.ttyc-1], ax ; al - set child's console tty
5811 <1> ; ah - reset child's wait channel
5812 0000E066 89F0 <1> mov eax, esi
5813 0000E068 A2[FB4E0100] <1> mov [u.uno], al ; child process number
5814 0000E06D FE86[F54B0100] <1> inc byte [esi+p.stat-1] ; 1, SRUN
5815 0000E073 66D1E6 <1> shl si, 1 ; multiply si by 2 to get index into p.pid table
5816 0000E076 66FF05[944E0100] <1> inc word [mpid] ; increment m.pid; get a new process name
5817 0000E07D 66A1[944E0100] <1> mov ax, [mpid]
5818 0000E083 668986[644B0100] <1> mov [esi+p.pid-2], ax ; put new process name
5819 <1> ; in child process' name slot
5820 <1> ;mov ax, [p.pid] ; get process name of MainProg
5821 0000E08A 66B80100 <1> mov ax, 1
5822 0000E08E 668986[844B0100] <1> mov [esi+p.ppid-2], ax ; put parent process name
5823 <1> ; in parent process slot for child

```

```

5824 0000E095 6648      <1>      dec     ax ; 0
5825 0000E097 66A3[DC4E0100] <1>      mov     [u.ttyp], ax ; 0
5826                                <1>      ;;
5827 0000E09D A1[994F0100] <1>      mov     eax, [ii]
5828                                <1>      ; Retro UNIX 386 v1, 'sysexec' (u2.s)
5829 0000E0A2 E819110000 <1>      call    iopen
5830                                <1>      ; 06/06/2016
5831 0000E0A7 C605[F14E0100]01 <1>      mov     byte [u.pri], 1 ; normal priority
5832                                <1>      ;
5833 0000E0AE EB0A      <1>      jmp      short sysexec_7 ; 02/05/2016
5834                                <1>
5835                                <1> sysexec_6:
5836                                <1>      ; 04/01/2017
5837                                <1>      ; 24/10/2016
5838                                <1>      ;;02/05/2016
5839                                <1>      ; 23/04/2016
5840                                <1>      ; 18/10/2015 ('sysexec_6')
5841                                <1>      ; 23/06/2015
5842 0000E0B0 A1[004F0100] <1>      mov     eax, [u.pgdir] ; physical address of page directory
5843                                <1>      ;;cmp eax, [k_page_dir] ; TRDOS MainProg ?
5844                                <1>      ;;je     short sysexec_7
5845 0000E0B5 E8F86BFFFF <1>      call    deallocate_page_dir
5846                                <1> sysexec_7:
5847 0000E0BA E8286BFFFF <1>      call    make_page_dir
5848 0000E0BF 0F824C82FFFF <1>      jc      panic ; allocation error
5849                                <1>      ; after a deallocation would be nonsense !?
5850                                <1>      ; 24/07/2015
5851                                <1>      ; map kernel pages (1st 4MB) to PDE 0
5852                                <1>      ; of the user's page directory
5853                                <1>      ; (It is needed for interrupts!)
5854                                <1>      ; 18/10/2015
5855 0000E0C5 8B15[20380100] <1>      mov     edx, [k_page_dir] ; Kernel's page directory
5856 0000E0CB 8B02      <1>      mov     eax, [edx] ; physical address of
5857                                <1>      ; kernel's first page table (1st 4 MB)
5858                                <1>      ; (PDE 0 of kernel's page directory)
5859 0000E0CD 8B15[004F0100] <1>      mov     edx, [u.pgdir]
5860 0000E0D3 8902      <1>      mov     [edx], eax ; PDE 0 (1st 4MB)
5861                                <1>      ;
5862                                <1>      ; 20/07/2015
5863 0000E0D5 BB00004000 <1>      mov     ebx, CORE ; start address = 0 (virtual) + CORE
5864                                <1>      ; 18/10/2015
5865 0000E0DA BE[844F0100] <1>      mov     esi, pcORE ; physical start address
5866                                <1> sysexec_8:
5867 0000E0DF B907000000 <1>      mov     ecx, PDE_A_USER + PDE_A_WRITE + PDE_A_PRESENT
5868 0000E0E4 E81C6BFFFF <1>      call    make_page_table
5869 0000E0E9 0F822282FFFF <1>      jc      panic
5870                                <1>      ;mov     ecx, PTE_A_USER + PTE_A_WRITE + PTE_A_PRESENT
5871 0000E0EF E81F6BFFFF <1>      call    make_page ; make new page, clear and set the pte
5872 0000E0F4 0F821782FFFF <1>      jc      panic
5873                                <1>      ;
5874 0000E0FA 8906      <1>      mov     [esi], eax ; 24/06/2015
5875                                <1>      ; ebx = virtual address (24/07/2015)
5876 0000E0FC E8B770FFFF <1>      call    add_to_swap_queue
5877                                <1>      ; 18/10/2015
5878 0000E101 81FE[884F0100] <1>      cmp     esi, ecore ; user's stack (last) page ?
5879 0000E107 740C      <1>      je      short sysexec_9 ; yes
5880 0000E109 BE[884F0100] <1>      mov     esi, ecore ; physical address of the last page
5881                                <1>      ; 20/07/2015
5882 0000E10E BB00F0FFFF <1>      mov     ebx, (ECORE - PAGE_SIZE) + CORE
5883                                <1>      ; ebx = virtual end address + segment base address - 4K
5884 0000E113 EB0A      <1>      jmp      short sysexec_8
5885                                <1> sysexec_9:
5886                                <1>      ; 24/04/2016
5887                                <1>      ; 18/10/2015
5888                                <1>      ; 26/08/2015
5889                                <1>      ; 25/06/2015
5890                                <1>      ; move arguments from kernel stack to [ecore]
5891                                <1>      ; (argument list/line will be copied from kernel stack
5892                                <1>      ; frame to the last (stack) page of user's core memory)
5893                                <1>      ; 18/10/2015
5894 0000E115 8B3D[884F0100] <1>      mov     edi, [ecore]
5895 0000E11B 81C700100000 <1>      add     edi, PAGE_SIZE
5896 0000E121 0FB705[924F0100] <1>      movzx   eax, word [argc]
5897 0000E128 09C0      <1>      or      eax, eax
5898 0000E12A 7509      <1>      jnz     short sysexec_10
5899 0000E12C 89FB      <1>      mov     ebx, edi
5900 0000E12E 83EB04 <1>      sub     ebx, 4
5901 0000E131 8903      <1>      mov     [ebx], eax ; 0
5902 0000E133 EB44      <1>      jmp      short sysexec_13
5903                                <1> sysexec_10:
5904 0000E135 8B0D[D44E0100] <1>      mov     ecx, [u.nread]
5905                                <1>      ;mov     esi, TextBuffer
5906 0000E13B 8B35[944F0100] <1>      mov     esi, [argv] ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)
5907 0000E141 29CF      <1>      sub     edi, ecx ; page end address - argument list length
5908 0000E143 89C2      <1>      mov     edx, eax
5909 0000E145 FEC2      <1>      inc     dl ; argument count + 1 for argc value
5910 0000E147 C0E202 <1>      shl     dl, 2 ; 4 * (argument count + 1)
5911 0000E14A 89FB      <1>      mov     ebx, edi
5912 0000E14C 80E3FC <1>      and     bl, 0FCh ; 32 bit (dword) alignment
5913 0000E14F 29D3      <1>      sub     ebx, edx
5914 0000E151 89FA      <1>      mov     edx, edi
5915 0000E153 F3A4      <1>      rep     movsb
5916 0000E155 89D6      <1>      mov     esi, edx
5917 0000E157 89DF      <1>      mov     edi, ebx
5918 0000E159 BA00F0BFFF <1>      mov     edx, ECORE - PAGE_SIZE ; virtual addr. of the last page
5919 0000E15E 2B15[884F0100] <1>      sub     edx, [ecore] ; difference (virtual - physical)
5920 0000E164 AB      <1>      stosd   ; eax = argument count
5921                                <1> sysexec_11:
5922 0000E165 89F0      <1>      mov     eax, esi
5923 0000E167 01D0      <1>      add     eax, edx
5924 0000E169 AB      <1>      stosd   ; eax = virtual address
5925 0000E16A FE0D[924F0100] <1>      dec     byte [argc]

```

```

5926 0000E170 7407      <1>      jz      short sysexec_13
5927                    <1> sysexec_12:
5928 0000E172 AC        <1>      lodsb
5929 0000E173 20C0      <1>      and     al, al
5930 0000E175 75FB      <1>      jnz     short sysexec_12
5931 0000E177 EBEC      <1>      jmp     short sysexec_11
5932                    <1> sysexec_13:
5933                    <1>      ; 24/10/2016
5934                    <1>      ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)
5935                    <1>      ; 23/06/2015 - 19/10/2015 (Retro UNIX 386 v1, 'sysexec_13')
5936                    <1>      ;
5937                    <1>      ; moving arguments to [ecore] is OK here..
5938                    <1>      ;
5939                    <1>      ; ebx = beginning addres of argument list pointers
5940                    <1>      ;      in user's stack
5941 0000E179 2B1D[884F0100] <1>      sub     ebx, [ecore]
5942 0000E17F 81C300F0BFFF <1>      add     ebx, (ECORE - PAGE_SIZE)
5943                    <1>      ; end of core - 4096 (last page)
5944                    <1>      ; (virtual address)
5945 0000E185 891D[944F0100] <1>      mov     [argv], ebx
5946 0000E18B 891D[D84E0100] <1>      mov     [u.break], ebx ; available user memory
5947                    <1>      ;
5948 0000E191 29C0      <1>      sub     eax, eax
5949 0000E193 C705[D04E0100]2000- <1>      mov     dword [u.count], 32 ; Executable file header size
5949 0000E19B 0000      <1>
5950 0000E19D C705[BC4E0100]- <1>      mov     dword [u.fofp], u.off
5950 0000E1A3 [C84E0100] <1>
5951 0000E1A7 A3[C84E0100] <1>      mov     [u.off], eax ; 0
5952 0000E1AC A3[CC4E0100] <1>      mov     [u.base], eax ; 0, start of user's core (virtual)
5953                    <1>      ; 24/10/2016
5954 0000E1B1 A0[E6380100] <1>      mov     al, [Current_Drv]
5955 0000E1B6 A2[8C4E0100] <1>      mov     [cdev], al
5956                    <1>      ;
5957 0000E1BB A1[994F0100] <1>      mov     eax, [ii] ; Fist Cluster of the Program (PRG) file
5958                    <1>      ; EAX = First cluster of the executable file
5959 0000E1C0 E832010000 <1>      call    readi
5960                    <1>
5961 0000E1C5 8B0D[D84E0100] <1>      mov     ecx, [u.break] ; top of user's stack (physical addr.)
5962 0000E1CB 890D[D04E0100] <1>      mov     [u.count], ecx ; save for overrun check
5963                    <1>      ;
5964 0000E1D1 8B0D[D44E0100] <1>      mov     ecx, [u.nread]
5965 0000E1D7 890D[D84E0100] <1>      mov     [u.break], ecx ; virtual address (offset from start)
5966 0000E1DD 80F920 <1>      cmp     cl, 32
5967 0000E1E0 7540 <1>      jne     short sysexec_15
5968                    <1>      ;:
5969                    <1>      ; Retro UNIX 386 v1 (32 bit) executable file header format
5970 0000E1E2 8B35[844F0100] <1>      mov     esi, [pcore] ; start address of user's core memory
5971                    <1>      ; (phys. start addr. of the exec. file)
5972 0000E1E8 AD <1>      lodsd
5973 0000E1E9 663DEB1E <1>      cmp     ax, 1EEBh ; EBH, 1Eh -> jump to +32
5974 0000E1ED 7533 <1>      jne     short sysexec_15
5975 0000E1EF AD <1>      lodsd
5976 0000E1F0 89C1 <1>      mov     ecx, eax ; text (code) section size
5977 0000E1F2 AD <1>      lodsd
5978 0000E1F3 01C1 <1>      add     ecx, eax ; + data section size (initialized data)
5979 0000E1F5 89CB <1>      mov     ebx, ecx
5980 0000E1F7 AD <1>      lodsd
5981 0000E1F8 01C3 <1>      add     ebx, eax ; + bss section size (for overrun checking)
5982 0000E1FA 3B1D[D04E0100] <1>      cmp     ebx, [u.count]
5983 0000E200 7711 <1>      ja     short sysexec_14 ; program overruns stack !
5984                    <1>      ;
5985                    <1>      ; add bss section size to [u.break]
5986 0000E202 0105[D84E0100] <1>      add     [u.break], eax
5987                    <1>      ;
5988 0000E208 83E920 <1>      sub     ecx, 32 ; header size (already loaded)
5989                    <1>      ;cmp    ecx, [u.count]
5990                    <1>      ;jnb    short sysexec_16
5991 0000E20B 890D[D04E0100] <1>      mov     [u.count], ecx ; required read count
5992 0000E211 EB29 <1>      jmp     short sysexec_16
5993                    <1> sysexec_14:
5994                    <1>      ; insufficient (out of) memory
5995 0000E213 C705[104F0100]0400- <1>      mov     dword [u.error], ERR_MINOR_IM ; 1
5995 0000E21B 0000 <1>
5996 0000E21D E9CDE1FFFF <1>      jmp     error
5997                    <1> sysexec_15:
5998 0000E222 8B15[9D4F0100] <1>      mov     edx, [i.size] ; file size
5999 0000E228 29CA <1>      sub     edx, ecx ; file size - loaded bytes
6000 0000E22A 7626 <1>      jna     short sysexec_17 ; no need to next read
6001 0000E22C 01D1 <1>      add     ecx, edx ; [i.size]
6002 0000E22E 3B0D[D04E0100] <1>      cmp     ecx, [u.count] ; overrun check (!)
6003 0000E234 77DD <1>      ja     short sysexec_14
6004 0000E236 8915[D04E0100] <1>      mov     [u.count], edx
6005                    <1> sysexec_16:
6006 0000E23C A1[994F0100] <1>      mov     eax, [ii] ; first cluster
6007 0000E241 E8B1000000 <1>      call    readi
6008 0000E246 8B0D[D44E0100] <1>      mov     ecx, [u.nread]
6009 0000E24C 010D[D84E0100] <1>      add     [u.break], ecx
6010                    <1> sysexec_17:
6011 0000E252 A1[994F0100] <1>      mov     eax, [ii] ; first cluster
6012 0000E257 E8650F0000 <1>      call    iclose
6013 0000E25C 31C0 <1>      xor     eax, eax
6014 0000E25E FEC0 <1>      inc     al
6015 0000E260 66A3[F24E0100] <1>      mov     [u.intr], ax ; 1 (interrupt/time-out is enabled)
6016 0000E266 66A3[F44E0100] <1>      mov     [u.quit], ax ; 1 ('crtl+brk' signal is enabled)
6017 0000E26C 833D[044F0100]00 <1>      cmp     dword [u.ppgdir], 0 ; is the caller MainProg (kernel) ?
6018 0000E273 770C <1>      ja     short sysexec_18 ; no, the caller is user process
6019                    <1>      ; If the caller is kernel (MainProg), 'sysexec' will come here
6020 0000E275 8B15[20380100] <1>      mov     edx, [k_page_dir] ; kernel's page directory
6021 0000E27B 8915[044F0100] <1>      mov     [u.ppgdir], edx ; next time 'sysexec' must not come here
6022                    <1> sysexec_18:
6023                    <1>      ; 02/05/2016
6024                    <1>      ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)

```

```

6025 <1> ; 18/10/2015 (Retro UNIX 386 v1)
6026 <1> ; 05/08/2015
6027 <1> ; 29/07/2015
6028 0000E281 8B2D[944F0100] <1> mov ebp, [argv] ; user's stack pointer must point to argument
6029 <1> ; list pointers (argument count)
6030 0000E287 FA <1> cli
6031 0000E288 8B25[BC370100] <1> mov esp, [tss.esp0] ; ring 0 (kernel) stack pointer
6032 <1> ;mov esp, [u.sp] ; Restore Kernel stack
6033 <1> ; for this process
6034 <1> ;add esp, 20 ; --> EIP, CS, EFLAGS, ESP, SS
6035 <1> ;xor eax, eax ; 0
6036 0000E28E FEC8 <1> dec al ; eax = 0
6037 0000E290 66BA2300 <1> mov dx, UDATA
6038 0000E294 6652 <1> push dx ; user's stack segment
6039 0000E296 55 <1> push ebp ; user's stack pointer
6040 <1> ; (points to number of arguments)
6041 <1>
6042 <1> ; 04/01/2017
6043 <1> ; MainProg comes here while [sysflg]= 0FFh
6044 <1> ; (but sysexec comes here while [sysflg]= 0)
6045 0000E297 C605[A14E0100]00 <1> mov byte [sysflg], 0 ; 04/01/2017
6046 <1> ; (timer_int sysflg control)
6047 0000E29E FB <1> sti
6048 0000E29F 9C <1> pushfd ; EFLAGS
6049 <1> ; Set IF for enabling interrupts in user mode
6050 <1> ;or dword [esp], 200h
6051 <1> ;
6052 <1> ;mov bx, UCODE
6053 <1> ;push bx ; user's code segment
6054 0000E2A0 6A1B <1> push UCODE
6055 <1> ;push 0
6056 0000E2A2 50 <1> push eax ; EIP (=0) - start address -
6057 0000E2A3 8925[A44E0100] <1> mov [u.sp], esp ; 29/07/2015
6058 <1> ; 05/08/2015
6059 <1> ; Remedy of a General Protection Fault during 'iretd' is here !
6060 <1> ; ('push dx' would cause to general protection fault,
6061 <1> ; after 'pop ds' etc.)
6062 <1> ;
6063 <1> ;; push dx ; ds (UDATA)
6064 <1> ;; push dx ; es (UDATA)
6065 <1> ;; push dx ; fs (UDATA)
6066 <1> ;; push dx ; gs (UDATA)
6067 <1> ;
6068 <1> ; This is a trick to prevent general protection fault
6069 <1> ; during 'iretd' intruction at the end of 'sysrele' (in ul.s):
6070 0000E2A9 8EC2 <1> mov es, dx ; UDATA
6071 0000E2AB 06 <1> push es ; ds (UDATA)
6072 0000E2AC 06 <1> push es ; es (UDATA)
6073 0000E2AD 06 <1> push es ; fs (UDATA)
6074 0000E2AE 06 <1> push es ; gs (UDATA)
6075 0000E2AF 66BA1000 <1> mov dx, KDATA
6076 0000E2B3 8EC2 <1> mov es, dx
6077 <1> ;
6078 <1> ;; pushad simulation
6079 0000E2B5 89E5 <1> mov ebp, esp ; esp before pushad
6080 0000E2B7 50 <1> push eax ; eax (0)
6081 0000E2B8 50 <1> push eax ; ecx (0)
6082 0000E2B9 50 <1> push eax ; edx (0)
6083 0000E2BA 50 <1> push eax ; ebx (0)
6084 0000E2BB 55 <1> push ebp ; esp before pushad
6085 0000E2BC 50 <1> push eax ; ebp (0)
6086 0000E2BD 50 <1> push eax ; esi (0)
6087 0000E2BE 50 <1> push eax ; edi (0)
6088 <1> ;
6089 0000E2BF A3[AC4E0100] <1> mov [u.r0], eax ; eax = 0
6090 0000E2C4 8925[A84E0100] <1> mov [u.usp], esp
6091 <1>
6092 <1> ; 02/05/2016
6093 <1> ;inc byte [sysflg] ; 0FFh -> 0
6094 <1> ;mov byte [sysflg], 0 ; 04/01/2017
6095 0000E2CA 0FB61D[FB4E0100] <1> movzx ebx, byte [u.uno]
6096 0000E2D1 6683BB[844B0100]01 <1> cmp word [ebx+p.ppid-2], 1 ; MainProg
6097 0000E2D9 0F8732E1FFFF <1> ja sysret0 ; 03/05/2016
6098 0000E2DF 68[0FC40000] <1> push sysret ; *
6099 0000E2E4 8925[A84E0100] <1> mov [u.usp], esp
6100 0000E2EA E886030000 <1> call wswap ; save child process 'u' structure and
6101 <1> ; registers
6102 0000E2EF 8305[A84E0100]04 <1> add dword [u.usp], 4 ; 03/05/2016
6103 <1> sysexec_19: ; 02/05/2016
6104 0000E2F6 C3 <1> retn ; * 'sysret' ; byte [sysflg] -> 0FFh
6105 <1>
6106 <1> readi:
6107 <1> ; 01/05/2016
6108 <1> ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
6109 <1> ; 20/05/2015 - Retro UNIX 386 v1
6110 <1> ; 11/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
6111 <1> ;
6112 <1> ; Reads from a file whose the first cluster number in EAX
6113 <1> ;
6114 <1> ; INPUTS ->
6115 <1> ; EAX - First cluster number of the file
6116 <1> ; u.count - byte count user desires
6117 <1> ; u.base - points to user buffer
6118 <1> ; u.fofp - points to dword with current file offset
6119 <1> ; i.size - file size
6120 <1> ; cdev - logical dos drive number of the file
6121 <1> ; OUTPUTS ->
6122 <1> ; u.count - cleared
6123 <1> ; u.nread - accumulates total bytes passed back
6124 <1> ;
6125 <1> ; ((EAX)) input/output
6126 <1> ; (Retro UNIX Prototype : 14/12/2012 - 01/03/2013, UNIXCOPY.ASM)

```



```

6127      <1>          ; ((Modified registers: edx, ebx, ecx, esi, edi))
6128      <1>
6129 0000E2F7 31D2      <1>          xor     edx, edx ; 0
6130 0000E2F9 8915[D44E0100] <1>          mov     [u.nread], edx ; 0
6131 0000E2FF 668915[0C4F0100] <1>          mov     [u.pcount], dx ; 19/05/2015
6132 0000E306 3915[D04E0100] <1>          cmp     [u.count], edx ; 0
6133 0000E30C 7701      <1>          ja      short readi_1
6134 0000E30E C3          <1>          retn
6135      <1> readi_1:
6136      <1> dskr:
6137      <1>          ; 01/05/2016
6138      <1>          ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
6139      <1>          ; 24/05/2015 - 12/10/2015 (Retro UNIX 386 v1)
6140      <1>          ; 26/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
6141      <1> dskr_0:
6142 0000E30F 8B15[9D4F0100] <1>          mov     edx, [i.size]
6143 0000E315 8B1D[BC4E0100] <1>          mov     ebx, [u.fofp]
6144 0000E31B 2B13      <1>          sub     edx, [ebx]
6145 0000E31D 7647      <1>          jna     short dskr_4
6146      <1>          ;
6147 0000E31F 50          <1>          push    eax ; 01/05/2016
6148 0000E320 3B15[D04E0100] <1>          cmp     edx, [u.count]
6149 0000E326 7306      <1>          jnb     short dskr_1
6150 0000E328 8915[D04E0100] <1>          mov     [u.count], edx
6151      <1> dskr_1:
6152      <1>          ; EAX = First Cluster
6153      <1>          ; [Current_Drv] = Physical drive number
6154 0000E32E E83B000000 <1>          call    mget_r
6155      <1>          ; NOTE: in 'mget_r', relevant sector will be read in buffer
6156      <1>          ; if it is not already in buffer !
6157 0000E333 BB[D4500100] <1>          mov     ebx, readi_buffer
6158 0000E338 803D[0E4F0100]00 <1>          cmp     byte [u.kcall], 0 ; the caller is 'namei' sign (=1)
6159 0000E33F 770F      <1>          ja      short dskr_3 ; zf=0 -> the caller is 'namei'
6160 0000E341 66833D[0C4F0100]00 <1>          cmp     word [u.pcount], 0
6161 0000E349 7705      <1>          ja      short dskr_3
6162      <1> dskr_2:
6163      <1>          ; [u.base] = virtual address to transfer (as destination address)
6164 0000E34B E894010000 <1>          call    trans_addr_w ; translate virtual address to physical (w)
6165      <1> dskr_3:
6166      <1>          ; EBX (r5) = system (I/O) buffer address -physical-
6167 0000E350 E8F7010000 <1>          call    sioereg
6168 0000E355 87F7      <1>          xchg    esi, edi
6169      <1>          ; EDI = file (user data) offset
6170      <1>          ; ESI = sector (I/O) buffer offset
6171      <1>          ; ECX = byte count
6172 0000E357 F3A4      <1>          rep     movsb
6173      <1>          ; eax = remain bytes in buffer
6174      <1>          ; (check if remain bytes in the buffer > [u.pcount])
6175 0000E359 09C0      <1>          or      eax, eax
6176 0000E35B 75EE      <1>          jnz     short dskr_2 ; (page end before system buffer end!)
6177 0000E35D 58          <1>          pop     eax ; (first cluster number)
6178 0000E35E 390D[D04E0100] <1>          cmp     [u.count], ecx ; 0
6179 0000E364 77A9      <1>          ja      short dskr_0
6180      <1> dskr_4:
6181 0000E366 C605[0E4F0100]00 <1>          mov     byte [u.kcall], 0
6182 0000E36D C3          <1>          retn
6183      <1>
6184      <1> mget_r:
6185      <1>          ; 24/10/2016
6186      <1>          ; 22/10/2016
6187      <1>          ; 12/10/2016
6188      <1>          ; 29/04/2016
6189      <1>          ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
6190      <1>          ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)
6191      <1>          ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
6192      <1>          ;
6193      <1>          ; Get existing or (allocate) a new disk block for file
6194      <1>          ;
6195      <1>          ; INPUTS ->
6196      <1>          ; [u.fofp] = file offset pointer
6197      <1>          ; EAX = First Cluster
6198      <1>          ; [cdev] = Logical dos drive number
6199      <1>          ; ([u.off] = file offset)
6200      <1>          ; OUTPUTS ->
6201      <1>          ; EAX = logical sector number
6202      <1>          ; ESI = Logical Dos Drive Description Table address
6203      <1>          ;
6204      <1>          ; Modified registers: EDX, EBX, ECX, ESI, EDI
6205      <1>
6206 0000E36E 8B35[BC4E0100] <1>          mov     esi, [u.fofp]
6207 0000E374 8B1E      <1>          mov     ebx, [esi] ; (u.off)
6208      <1>
6209 0000E376 29C9      <1>          sub     ecx, ecx
6210 0000E378 8A2D[8C4E0100] <1>          mov     ch, [cdev]
6211      <1>
6212 0000E37E BE00010900 <1>          mov     esi, Logical_DOSDisks
6213 0000E383 01CE      <1>          add     esi, ecx
6214      <1>
6215 0000E385 380D[58450100] <1>          cmp     [readi.valid], cl ; 0
6216 0000E38B 7649      <1>          jna     short mget_r_0
6217      <1>
6218 0000E38D 3A2D[59450100] <1>          cmp     ch, [readi.driv]
6219 0000E393 7541      <1>          jne     short mget_r_0
6220      <1>
6221 0000E395 3B05[6C450100] <1>          cmp     eax, [readi.fclust]
6222 0000E39B 7565      <1>          jne     short mget_r_3
6223      <1>
6224 0000E39D 89D8      <1>          mov     eax, ebx ; file offset
6225 0000E39F 668B0D[60450100] <1>          mov     cx, [readi.bpc]
6226 0000E3A6 41          <1>          inc     ecx ; <= 65536
6227 0000E3A7 29D2      <1>          sub     edx, edx
6228 0000E3A9 F7F1      <1>          div     ecx

```

6229		<1>	
6230	0000E3AB 8B3D[68450100]	<1>	mov edi,[readi.c_index] ; cluster index
6231		<1>	
6232	0000E3B1 39F8	<1>	cmp eax, edi
6233	0000E3B3 757A	<1>	jne short mget_r_4 ; (*)
6234		<1>	
6235		<1>	; edx = byte offset in cluster (<= 65535)
6236	0000E3B5 668915[62450100]	<1>	mov [readi.offset], dx
6237	0000E3BC 66C1EA09	<1>	shr dx, 9 ; / 512
6238	0000E3C0 8815[5B450100]	<1>	mov [readi.s_index], dl ; sector index in cluster (0 to spc -1)
6239		<1>	
6240	0000E3C6 A1[64450100]	<1>	mov eax, [readi.cluster] ; > 0 if [readi.valid] = 1
6241	0000E3CB 8B15[70450100]	<1>	mov edx, [readi.fs_index]
6242	0000E3D1 E99A000000	<1>	jmp mget_r_7
6243		<1>	
6244		<1>	mget_r_0:
6245	0000E3D6 882D[59450100]	<1>	mov [readi.driv], ch ; physical drive number
6246	0000E3DC 807E0300	<1>	cmp byte [esi+LD_FATType], 0
6247	0000E3E0 7707	<1>	ja short mget_r_1
6248	0000E3E2 8A4E12	<1>	mov cl, [esi+LD_FS_BytesPerSec+1]
6249	0000E3E5 D0E9	<1>	shr cl, 1 ; ; 1 for 512 bytes, 4 for 2048 bytes
6250	0000E3E7 EB03	<1>	jmp short mget_r_2
6251		<1>	mget_r_1:
6252	0000E3E9 8A4E13	<1>	mov cl, [esi+LD_BPB+BPB_SecPerClust]
6253		<1>	mget_r_2:
6254	0000E3EC 880D[5A450100]	<1>	mov [readi.spc], cl ; sectors per cluster
6255		<1>	; NOTE: readi bytes per sector value is always 512 !
6256	0000E3F2 66C1E109	<1>	shl cx, 9 ; * 512
6257	0000E3F6 6649	<1>	dec cx ; bytes per cluster - 1
6258	0000E3F8 66890D[60450100]	<1>	mov [readi.bpc], cx
6259	0000E3FF 6629C9	<1>	sub cx, cx
6260		<1>	mget_r_3:
6261	0000E402 A3[6C450100]	<1>	mov [readi.fclust], eax ; first cluster (or FDT address)
6262	0000E407 880D[58450100]	<1>	mov [readi.valid], cl ; 0
6263		<1>	;mov [readi.s_index], cl ; 0
6264		<1>	;mov [readi.offset], cx ; 0
6265	0000E40D 890D[68450100]	<1>	mov [readi.c_index], ecx ; 0
6266	0000E413 890D[64450100]	<1>	mov [readi.cluster], ecx ; 0
6267	0000E419 890D[5C450100]	<1>	mov [readi.sector], ecx ; 0
6268		<1>	
6269	0000E41F 89D8	<1>	mov eax, ebx ; file offset
6270	0000E421 668B0D[60450100]	<1>	mov cx, [readi.bpc]
6271	0000E428 41	<1>	inc ecx ; <= 65536
6272	0000E429 29D2	<1>	sub edx, edx
6273	0000E42B F7F1	<1>	div ecx
6274		<1>	;mov edi, [readi.c_index] ; previous cluster index
6275	0000E42D 29FF	<1>	sub edi, edi
6276		<1>	mget_r_4:
6277	0000E42F A3[68450100]	<1>	mov [readi.c_index], eax ; cluster index
6278		<1>	; edx = byte offset in cluster (<= 65535)
6279	0000E434 668915[62450100]	<1>	mov [readi.offset], dx
6280	0000E43B 66C1EA09	<1>	shr dx, 9 ; / 512
6281	0000E43F 8815[5B450100]	<1>	mov [readi.s_index], dl ; sector index in cluster (0 to spc -1)
6282		<1>	
6283	0000E445 89C1	<1>	mov ecx, eax ; current cluster index
6284	0000E447 A1[6C450100]	<1>	mov eax, [readi.fclust]
6285	0000E44C 09C9	<1>	or ecx, ecx ; cluster index
6286	0000E44E 741B	<1>	jz short mget_r_6
6287		<1>	
6288	0000E450 39CF	<1>	cmp edi, ecx
6289	0000E452 7710	<1>	ja short mget_r_5 ; old cluster index is higher
6290	0000E454 8B15[64450100]	<1>	mov edx, [readi.cluster]
6291	0000E45A 21D2	<1>	and edx, edx
6292	0000E45C 7406	<1>	jz short mget_r_5
6293		<1>	; valid 'readi' parameters (*)
6294	0000E45E 89D0	<1>	mov eax, edx
6295	0000E460 29F9	<1>	sub ecx, edi
6296	0000E462 740C	<1>	jz short mget_r_7
6297		<1>	mget_r_5:
6298		<1>	; EAX = Beginning cluster
6299		<1>	; EDX = Sector index in disk/file section
6300		<1>	; (Only for SINGLIX file system!)
6301		<1>	; ECX = Cluster sequence number after the beginning cluster
6302		<1>	; ESI = Logical DOS Drive Description Table address
6303	0000E464 E823DEFFFF	<1>	call get_cluster_by_index
6304	0000E469 724E	<1>	jc short mget_r_err
6305		<1>	; EAX = Cluster number
6306		<1>	mget_r_6:
6307	0000E46B A3[64450100]	<1>	mov [readi.cluster], eax ; FDT number for Singlix File System
6308			

```

6331      <1>
6332      <1>      ; EAX = Logical sector number
6333      <1>      ; ECX = Sector count
6334      <1>      ; EBX = Buffer address
6335      <1>      ; (EDX = 0)
6336      <1>      ; ESI = Logical DOS drive description table address
6337      <1>
6338      0000E4AD E8250D0000      <1>      call    disk_read
6339      0000E4B2 7314           <1>      jnc     short mget_r_10
6340      <1>
6341      <1>      ; 22/10/2016 (15h -> 17)
6342      0000E4B4 B811000000      <1>      mov     eax, 17 ; Drive not ready or read error !
6343      <1> mget_r_err:          <1>
6344      0000E4B9 A3[104F0100]    <1>      mov     [u.error], eax
6345      <1>      ; 12/10/2016
6346      0000E4BE A3[AC4E0100]    <1>      mov     [u.r0], eax
6347      0000E4C3 E927DFFFFFFF    <1>      jmp     error
6348      <1> mget_r_10:          <1>
6349      0000E4C8 C605[58450100]01 <1>      mov     byte [readi.valid], 1 ; 24/10/2016
6350      0000E4CF A1[5C450100]    <1>      mov     eax, [readi.sector]
6351      <1> mget_r_11:          <1>
6352      0000E4D4 C3              <1>      retn
6353      <1> mget_r_12:          <1>
6354      <1>      ; EAX = FDT number
6355      <1>      ; EDX = Sector index from FDT sector (0,1,2,3,4...)
6356      0000E4D5 40              <1>      inc     eax ; the first data sector in FS disk section
6357      0000E4D6 8915[70450100]    <1>      mov     [readi.fs_index], edx
6358      0000E4DC 01D0           <1>      add     eax, edx
6359      0000E4DE EBAD           <1>      jmp     short mget_r_8
6360      <1>
6361      <1> trans_addr_r:        <1>
6362      <1>      ; 12/10/2016
6363      <1>      ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)
6364      <1>      ; Translate virtual address to physical address
6365      <1>      ; for reading from user's memory space
6366      <1>      ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)
6367      <1>
6368      0000E4E0 31D2           <1>      xor     edx, edx ; 0 (read access sign)
6369      0000E4E2 EB04           <1>      jmp     short trans_addr_rw
6370      <1>
6371      <1> trans_addr_w:        <1>
6372      <1>      ; 12/10/2016
6373      <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6374      <1>      ; Translate virtual address to physical address
6375      <1>      ; for writing to user's memory space
6376      <1>      ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)
6377      <1>
6378      0000E4E4 29D2           <1>      sub     edx, edx
6379      0000E4E6 FEC2           <1>      inc     dl ; 1 (write access sign)
6380      <1> trans_addr_rw:        <1>
6381      0000E4E8 50              <1>      push    eax
6382      0000E4E9 53              <1>      push    ebx
6383      0000E4EA 52              <1>      push    edx ; r/w sign (in DL)
6384      <1>      ;
6385      0000E4EB 8B1D[CC4E0100]    <1>      mov     ebx, [u.base]
6386      0000E4F1 E8986DFFFFFF    <1>      call    get_physical_addr ; get physical address
6387      0000E4F6 730F           <1>      jnc     short passc_0
6388      0000E4F8 A3[104F0100]    <1>      mov     [u.error], eax
6389      0000E4FD A3[AC4E0100]    <1>      mov     [u.r0], eax ; 12/10/2016
6390      <1>      ;pop    edx
6391      <1>      ;pop    ebx
6392      <1>      ;pop    eax
6393      0000E502 E9E8DEFFFFFF    <1>      jmp     error
6394      <1> passc_0:              <1>
6395      0000E507 F6C202          <1>      test    dl, PTE_A_WRITE ; writable page
6396      0000E50A 5A              <1>      pop     edx
6397      0000E50B 751C           <1>      jnz     short passc_1
6398      <1>
6399      0000E50D 20D2           <1>      and     dl, dl
6400      0000E50F 7418           <1>      jz      short passc_1
6401      <1>      ; read only (duplicated) page -must be copied to a new page-
6402      <1>      ; EBX = linear address
6403      0000E511 51              <1>      push    ecx
6404      0000E512 E8106AFFFF    <1>      call    copy_page
6405      0000E517 59              <1>      pop     ecx
6406      0000E518 721E           <1>      jc      short passc_2
6407      0000E51A 50              <1>      push    eax ; physical address of the new/allocated page
6408      0000E51B E8986CFFFF    <1>      call    add_to_swap_queue
6409      0000E520 58              <1>      pop     eax
6410      0000E521 81E3FF0F0000    <1>      and     ebx, PAGE_OFF ; 0FFFh
6411      <1>      ;mov    ecx, PAGE_SIZE
6412      <1>      ;sub     ecx, ebx
6413      0000E527 01D8           <1>      add     eax, ebx
6414      <1> passc_1:              <1>
6415      0000E529 A3[084F0100]    <1>      mov     [u.pbase], eax ; physical address
6416      0000E52E 66890D[0C4F0100] <1>      mov     [u.pcount], cx ; remain byte count in page (1-4096)
6417      0000E535 5B              <1>      pop     ebx
6418      0000E536 58              <1>      pop     eax
6419      0000E537 C3              <1>      retn
6420      <1> passc_2:              <1>
6421      0000E538 B804000000    <1>      mov     eax, ERR_MINOR_IM ; "Insufficient memory !" error
6422      0000E53D A3[AC4E0100]    <1>      mov     [u.r0], eax ; 12/10/2016
6423      0000E542 A3[104F0100]    <1>      mov     dword [u.error], eax
6424      <1>      ;pop    ebx
6425      <1>      ;pop    eax
6426      0000E547 E9A3DEFFFFFF    <1>      jmp     error
6427      <1>
6428      <1> sioreg:              <1>
6429      <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6430      <1>      ; 19/05/2015 - 25/07/2015 (Retro UNIX 386 v1)
6431      <1>      ; 12/03/2013 - 22/07/2013 (Retro UNIX 8086 v1)
6432      <1>      ; INPUTS ->

```

```

6433 <1> ; EBX = system buffer (data) address (r5)
6434 <1> ; [u.fofp] = pointer to file offset pointer
6435 <1> ; [u.base] = virtual address of the user buffer
6436 <1> ; [u.pbase] = physical address of the user buffer
6437 <1> ; [u.count] = byte count
6438 <1> ; [u.pcount] = byte count within page frame
6439 <1> ; OUTPUTS ->
6440 <1> ; ESI = user data offset (r1)
6441 <1> ; EDI = system (I/O) buffer offset (r2)
6442 <1> ; ECX = byte count (r3)
6443 <1> ; EAX = remain bytes after byte count within page frame
6444 <1> ; (If EAX > 0, transfer will continue from the next page)
6445 <1> ;
6446 <1> ; ((Modified registers: EDX))
6447 <1>
6448 0000E54C 8B35[BC4E0100] <1> mov esi, [u.fofp]
6449 0000E552 8B3E <1> mov edi, [esi]
6450 0000E554 89F9 <1> mov ecx, edi
6451 0000E556 81C900FFFFFF <1> or ecx, 0FFFFFFE00h
6452 0000E55C 81E7FF010000 <1> and edi, 1FFh
6453 0000E562 01DF <1> add edi, ebx ; EBX = system buffer (data) address
6454 0000E564 F7D9 <1> neg ecx
6455 0000E566 3B0D[D04E0100] <1> cmp ecx, [u.count]
6456 0000E56C 7606 <1> jna short sioreg_0
6457 0000E56E 8B0D[D04E0100] <1> mov ecx, [u.count]
6458 <1> sioreg_0:
6459 0000E574 803D[0E4F0100]00 <1> cmp byte [u.kcall], 0
6460 0000E57B 7613 <1> jna short sioreg_1
6461 <1> ; the caller is 'mkdir' or 'namei'
6462 0000E57D A1[CC4E0100] <1> mov eax, [u.base]
6463 0000E582 A3[084F0100] <1> mov [u.pbase], eax ; physical address = virtual address
6464 0000E587 66890D[0C4F0100] <1> mov word [u.pcount], cx ; remain bytes in buffer (1 sector)
6465 0000E58E EB0B <1> jmp short sioreg_2
6466 <1> sioreg_1:
6467 0000E590 0FB715[0C4F0100] <1> movzx edx, word [u.pcount]
6468 0000E597 39D1 <1> cmp ecx, edx
6469 0000E599 772A <1> ja short sioreg_4 ; transfer count > [u.pcount]
6470 <1> sioreg_2: ; 2:
6471 0000E59B 31C0 <1> xor eax, eax
6472 <1> sioreg_3:
6473 0000E59D 010D[D44E0100] <1> add [u.nread], ecx
6474 0000E5A3 290D[D04E0100] <1> sub [u.count], ecx
6475 0000E5A9 010D[CC4E0100] <1> add [u.base], ecx
6476 0000E5AF 010E <1> add [esi], ecx
6477 0000E5B1 8B35[084F0100] <1> mov esi, [u.pbase]
6478 0000E5B7 66290D[0C4F0100] <1> sub [u.pcount], cx
6479 0000E5BE 010D[084F0100] <1> add [u.pbase], ecx
6480 0000E5C4 C3 <1> retn
6481 <1> sioreg_4:
6482 <1> ; transfer count > [u.pcount]
6483 <1> ; (ecx > edx)
6484 0000E5C5 89C8 <1> mov eax, ecx
6485 0000E5C7 29D0 <1> sub eax, edx ; remain bytes for 1 sector (block) transfer
6486 0000E5C9 89D1 <1> mov ecx, edx ; current transfer count = [u.pcount]
6487 0000E5CB EBD0 <1> jmp short sioreg_3
6488 <1>
6489 <1> tswitch: ; Retro UNIX 386 v1
6490 <1> tswap:
6491 <1> ; 16/01/2017
6492 <1> ; 21/05/2016 - TRDOS 386 (TRDOS v2.0)
6493 <1> ; 10/05/2015 - 01/09/2015 (Retro UNIX 386 v1)
6494 <1> ; 14/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
6495 <1> ; time out swap, called when a user times out.
6496 <1> ; the user is put on the low priority queue.
6497 <1> ; This is done by making a link from the last user
6498 <1> ; on the low priority queue to him via a call to 'putlu'.
6499 <1> ; then he is swapped out.
6500 <1>
6501 <1> ; TRDOS 386 (TRDOS v2.0) modification -> ** 21/05/2016 **
6502 <1> ; * when a high priority (event) process will be stopped
6503 <1> ; (swapped out, switched out/off), 'tswap/tswitch' will
6504 <1> ; not add it to a run queue.
6505 <1> ; /// What for: Process may be already in a run queue,
6506 <1> ; it is unspecied state because process might be started
6507 <1> ; by a timer event which does not regard previous priority
6508 <1> ; level and run queue of the process (for fast executing!).
6509 <1> ; After the 'run for event', process will be sequenced
6510 <1> ; to run by it's actual run queue. ///
6511 <1> ;
6512 <1> ; Retro UNIX 386 v1 modification ->
6513 <1> ; swap (software task switch) is performed by changing
6514 <1> ; user's page directory (u.pgdir) instead of segment change
6515 <1> ; as in Retro UNIX 8086 v1.
6516 <1> ;
6517 <1> ; RETRO UNIX 8086 v1 modification ->
6518 <1> ; 'swap to disk' is replaced with 'change running segment'
6519 <1> ; according to 8086 cpu (x86 real mode) architecture.
6520 <1> ; pdp-11 was using 64KB uniform memory while IBM PC
6521 <1> ; compatibles was using 1MB segmented memory
6522 <1> ; in 8086/8088 times.
6523 <1> ;
6524 <1> ; INPUTS ->
6525 <1> ; u.uno - users process number
6526 <1> ; runq+4 - lowest priority queue
6527 <1> ; OUTPUTS ->
6528 <1> ; r0 - users process number
6529 <1> ; r2 - lowest priority queue address
6530 <1> ;
6531 <1> ; ((AX = R0, BX = R2)) output
6532 <1> ; ((Modified registers: EDX, EBX, ECX, ESI, EDI))
6533 <1> ;
6534 <1>

```



```

6535 <1> NOTE:
6536 <1> ;* [u.pri] priority level is specified by run queue which is process
6537 <1> ; comes to run from.
6538 <1> ;* Initial [u.pri] is 1 ('normal/regular') for programs
6539 <1> ; (which are launched by MainProg or 'sysexec'), it is changed
6540 <1> ; to 2 ('high') by timer event, if program uses 'systemtimer' system call.
6541 <1> ;* Program (Process) also can change it's running priority
6542 <1> ; from 1 to 0 or up to 2 by using 'syspri' system call; but,
6543 <1> ; if program selects priority level 2 (high) for running, next time
6544 <1> ; it is reduced to 1 (normal/regular) because 'syspri' adds this
6545 <1> ; program to 'run for normal' queue while running duration is a bit
6546 <1> ; protected from swap/switch out immediate, behalf of other high
6547 <1> ; priority process in sequence. Program (with high priority) will not
6548 <1> ; be swapped/switched out (by timer event) before it's time quantum
6549 <1> ; will be elapsed, but, this will be temporary if program is not using
6550 <1> ; timer event function.
6551 <1>
6552 <1> ;For example:
6553 <1> ;If a process frequently gets a timer event, it runs at high priority
6554 <1> ;level but when it returns from running it returns to actual run queue,
6555 <1> ;not to 'run for event' queue again.
6556 <1> ;'tswap' will not change the sequence at return/stop(swap out) stage.
6557 <1> ;But if priority level not high (=2, 'run for event'), 'tswap/tswitch'
6558 <1> ;will add the stopping process to relevant run queue according to
6559 <1> ;[u.pri] priority level.
6560 <1>
6561 <1> ; 16/01/2017
6562 0000E5CD BB[9A4E0100] <1> mov ebx, runq+2 ; 'runq_normal' ; normal/regular priority
6563 <1> ; 21/05/2016
6564 <1> ;cmp byte [u.pri], 2 ; high priority (run for event) ?
6565 <1> ;jnb short swap
6566 <1> ; 16/01/2017
6567 <1> ; (Normal and also high/event priority processes will be added to
6568 <1> ; normal priority run queue for ensuring circular running sequence!)
6569 <1> ; (Timer interrupt or 'syspri' system call may change priority and run
6570 <1> ; queue to high/event level.)
6571 0000E5D2 803D[F14E0100]00 <1> cmp byte [u.pri], 0
6572 0000E5D9 7702 <1> ja short tswap_1; normal priority run queue
6573 <1> ;
6574 0000E5DB 43 <1> inc ebx
6575 0000E5DC 43 <1> inc ebx ; runq+4, 'runq_background', low priority
6576 <1> tswap_1:
6577 0000E5DD A0[FB4E0100] <1> mov al, [u.uno]
6578 <1> ; movb u.uno,r1 / move users process number to r1
6579 <1> ; mov $runq+4,r2
6580 <1> ; / move lowest priority queue address to r2
6581 <1> ; ebx = run queue
6582 0000E5E2 E8FE000000 <1> call putlu
6583 <1> ; jsr r0,putlu / create link from last user on Q to
6584 <1> ; / u.uno's user
6585 <1>
6586 <1> switch: ; Retro UNIX 386 v1
6587 <1> swap:
6588 <1> ; 02/01/2017
6589 <1> ; 21/05/2016
6590 <1> ; 20/05/2016
6591 <1> ; 02/05/2016
6592 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6593 <1> ; 10/05/2015 - 02/09/2015 (Retro UNIX 386 v1)
6594 <1> ; 14/04/2013 - 08/03/2014 (Retro UNIX 8086 v1)
6595 <1> ;
6596 <1> ; 'swap' is routine that controls the swapping of processes
6597 <1> ; in and out of core.
6598 <1> ;
6599 <1> ; TRDOS 386 (TRDOS v2.0) modification -> ** 20/05/2016 **
6600 <1> ; * 3 different priority level is applied
6601 <1> ; (just as original unix v1)
6602 <1> ; 1) high priority (event) run queue, 'runq_event'
6603 <1> ; 2) normal priority (regular) run queue, 'runq_normal'
6604 <1> ; 3) low priority (background) run queue, 'runq_backgroud'
6605 <1> ; 'swap' code will run a process which has max. priority
6606 <1> ; (for earliest event at first)
6607 <1> ;
6608 <1> ; Retro UNIX 386 v1 modification ->
6609 <1> ; swap (software task switch) is performed by changing
6610 <1> ; user's page directory (u.pgdir) instead of segment change
6611 <1> ; as in Retro UNIX 8086 v1.
6612 <1> ;
6613 <1> ; RETRO UNIX 8086 v1 modification ->
6614 <1> ; 'swap to disk' is replaced with 'change running segment'
6615 <1> ; according to 8086 cpu (x86 real mode) architecture.
6616 <1> ; pdp-11 was using 64KB uniform memory while IBM PC
6617 <1> ; compatibles was using 1MB segmented memory
6618 <1> ; in 8086/8088 times.
6619 <1> ;
6620 <1> ; INPUTS ->
6621 <1> ; runq table - contains processes to run.
6622 <1> ; p.link - contains next process in line to be run.
6623 <1> ; u.uno - process number of process in core
6624 <1> ; s.stack - swap stack used as an internal stack for swapping.
6625 <1> ; OUTPUTS ->
6626 <1> ; (original unix v1 -> present process to its disk block)
6627 <1> ; (original unix v1 -> new process into core ->
6628 <1> ; Retro Unix 8086 v1 -> segment registers changed
6629 <1> ; for new process)
6630 <1> ; u.quant = 3 (Time quantum for a process)
6631 <1> ; ((INT 1Ch count down speed -> 18.2 times per second)
6632 <1> ; RETRO UNIX 8086 v1 will use INT 1Ch (18.2 times per second)
6633 <1> ; for now, it will swap the process if there is not
6634 <1> ; a keyboard event (keystroke) (Int 15h, function 4Fh)
6635 <1> ; or will count down from 3 to 0 even if there is a
6636 <1> ; keyboard event locking due to repetitive key strokes.

```

```

6637      <1>      ;      u.quant will be reset to 3 for RETRO UNIX 8086 v1.
6638      <1>      ;
6639      <1>      ; ((Modified registers: EAX, EDX, EBX, ECX, ESI, EDI))
6640      <1>
6641      <1>      ;NOTE:
6642      <1>      ;High priority queue is the first for selecting a process to run.
6643      <1>      ;If there is not a process in high priority level run queue,
6644      <1>      ;a process in normal priority run queue will be selected
6645      <1>      ;or a proces in low priority run queue will be selected if normal
6646      <1>      ;priority level run queue is empty.
6647      <1>
6648      <1>      ; 21/05/2016 -(3 priority levels, 3 run queues)
6649 0000E5E7 BE[984E0100] <1>      mov     esi, runq ; 'runq_event' ; high priority, 'run for event'
6650 0000E5EC C605[B4450100]03 <1>      mov     byte [priority], 3 ; high priority + 1
6651 0000E5F3 31DB <1>      xor     ebx, ebx ; 02/01/2017
6652 <1> swap_0: ; 1: / search runq table for highest priority process
6653 0000E5F5 66AD <1>      lodsw   ; mov ax, [esi], add esi+2
6654 <1>      ;xor     ebx, ebx ; 02/05/2016
6655 0000E5F7 6621C0 <1>      and     ax, ax ; are there any processes to run in this Q entry
6656 0000E5FA 750E <1>      jnz     short swap_2
6657 <1>      ; 21/05/2026
6658 <1>      ; runq_normal = runq+2, runq_background = runq+4
6659 0000E5FC FE0D[B4450100] <1>      dec     byte [priority] ; 3 -> 3, 2 -> 1, 1-> 0
6660 0000E602 75F1 <1>      jnz     short swap_0
6661 <1>      ;cmp     esi, runq+6 ; if zero compare address to end of table
6662 <1>      ;jb     short swap_0 ; if not at end, go back
6663 <1> swap_1:
6664 <1>      ; 02/05/2016
6665 <1>      ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
6666 <1>      ; No user process to run...
6667 <1>      ; Run the kernel process... MainProg: Internal Command Interpreter
6668 0000E604 FEC0 <1>      inc     al, mov al, 1 ; process number of MainProg
6669 0000E606 FEC3 <1>      inc     bl ; mov bl, al ; 1
6670 0000E608 EB1E <1>      jmp     short swap_4
6671 <1> swap_2:
6672 <1>      ; 21/05/2016
6673 0000E60A FE0D[B4450100] <1>      dec     byte [priority] ; priority level of present user/process
6674 <1>      ; 0, 1, 2
6675 0000E610 4E <1>      dec     esi
6676 0000E611 4E <1>      dec     esi
6677 <1>      ;
6678 0000E612 88C3 <1>      mov     bl, al
6679 0000E614 38E0 <1>      cmp     al, ah ; is there only 1 process in the queue to be run
6680 0000E616 740A <1>      je      short swap_3 ; yes
6681 0000E618 8AA3[E54B0100] <1>      mov     ah, [ebx+p.link-1]
6682 0000E61E 8826 <1>      mov     [esi], ah ; move next process in line into run queue
6683 0000E620 EB06 <1>      jmp     short swap_4
6684 <1> swap_3:
6685 0000E622 6631D2 <1>      xor     dx, dx
6686 0000E625 668916 <1>      mov     [esi], dx ; zero the entry; no processes on the Q
6687 <1> swap_4:
6688 0000E628 8A25[FB4E0100] <1>      mov     ah, [u.uno]
6689 0000E62E 38C4 <1>      cmp     ah, al ;is this process the same as the process in core?
6690 0000E630 743B <1>      je      short swap_8 ; yes, don't have to swap
6691 0000E632 08E4 <1>      or      ah, ah ; is the process # = 0
6692 0000E634 740D <1>      jz      short swap_6 ; 'sysexit'
6693 <1>      ;cmp     ah, al ;is this process the same as the process in core?
6694 <1>      ;je      short swap_8 ; yes, don't have to swap
6695 0000E636 8925[A84E0100] <1>      mov     [u.usp], esp ; return address for 'syswait' & 'sleep'
6696 0000E63C E834000000 <1>      call    wswap ; write out core to disk
6697 0000E641 EB1C <1>      jmp     short swap_7
6698 <1> swap_6:
6699 <1>      ; Deallocate memory pages belong to the process
6700 <1>      ; which is being terminated.
6701 <1>      ; (Retro UNIX 386 v1 modification !)
6702 <1>      ;
6703 0000E643 53 <1>      push    ebx
6704 0000E644 A1[004F0100] <1>      mov     eax, [u.pgdir] ; page directory of the process
6705 0000E649 8B1D[044F0100] <1>      mov     ebx, [u.ppgdir] ; page directory of the parent process
6706 0000E64F E85E66FFFF <1>      call    deallocate_page_dir
6707 0000E654 A1[FC4E0100] <1>      mov     eax, [u.upage] ; 'user' structure page of the process
6708 0000E659 E8F966FFFF <1>      call    deallocate_page
6709 0000E65E 5B <1>      pop     ebx
6710 <1> swap_7:
6711 0000E65F C0E302 <1>      shl     bl, 2 ; * 4
6712 0000E662 8B83[024C0100] <1>      mov     eax, [ebx+p.upage-4] ; the 'u' page of the new process
6713 0000E668 E840000000 <1>      call    rswap ; read new process into core
6714 <1> swap_8:
6715 <1>      ; Retro UNIX 8086 v1 modification !
6716 0000E66D C605[F04E0100]04 <1>      mov     byte [u.quant], time_count
6717 0000E674 C3 <1>      retn
6718 <1>
6719 <1> wswap: ; < swap out, swap to disk >
6720 <1>      ; 28/02/2017 (fnsave)
6721 <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6722 <1>      ; 09/05/2015 (Retro UNIX 386 v1)
6723 <1>      ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
6724 <1>      ; 'wswap' writes out the process that is in core onto its
6725 <1>      ; appropriate disk area.
6726 <1>      ;
6727 <1>      ; Retro UNIX 386 v1 modification ->
6728 <1>      ;      User (u) structure content and the user's register content
6729 <1>      ;      will be copied to the process's/user's UPAGE (a page for
6730 <1>      ;      saving 'u' structure and user registers for task switching).
6731 <1>      ;      u.usp - points to kernel stack address which contains
6732 <1>      ;      user's registers while entering system call.
6733 <1>      ;      u.sp - points to kernel stack address
6734 <1>      ;      to return from system call -for IRET-.
6735 <1>      ;      [u.usp]+32+16 = [u.sp]
6736 <1>      ;      [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
6737 <1>      ;      edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
6738 <1>      ;

```

```

6739 <1> ; Retro UNIX 8086 v1 modification ->
6740 <1> ; 'swap to disk' is replaced with 'change running segment'
6741 <1> ; according to 8086 cpu (x86 real mode) architecture.
6742 <1> ; pdp-11 was using 64KB uniform memory while IBM PC
6743 <1> ; compatibles was using 1MB segmented memory
6744 <1> ; in 8086/8088 times.
6745 <1> ;
6746 <1> ; INPUTS ->
6747 <1> ; u.break - points to end of program
6748 <1> ; u.usp - stack pointer at the moment of swap
6749 <1> ; core - beginning of process program
6750 <1> ; ecore - end of core
6751 <1> ; user - start of user parameter area
6752 <1> ; u.uno - user process number
6753 <1> ; p.dska - holds block number of process
6754 <1> ; OUTPUTS ->
6755 <1> ; swp I/O queue
6756 <1> ; p.break - negative word count of process
6757 <1> ; r1 - process disk address
6758 <1> ; r2 - negative word count
6759 <1> ;
6760 <1> ; RETRO UNIX 8086 v1 input/output:
6761 <1> ;
6762 <1> ; INPUTS ->
6763 <1> ; u.uno - process number (to be swapped out)
6764 <1> ; OUTPUTS ->
6765 <1> ; none
6766 <1> ;
6767 <1> ; ((Modified registers: ECX, ESI, EDI))
6768 <1> ;
6769 <1> ;
6770 <1> ; 28/02/2017
6771 <1> ; cmp byte [multi_tasking], 0 ; Musti tasking mode ?
6772 <1> ; jna short wswp
6773 0000E675 803D[224F0100]00 <1> cmp byte [u.fpsave], 0 ; 28/02/2017
6774 0000E67C 7606 <1> jna short wswp
6775 0000E67E DD35[244F0100] <1> fnsave [u.fpregs] ; save floating point registers (94 bytes)
6776 <1> wswp:
6777 0000E684 8B3D[FC4E0100] <1> mov edi, [u.upage] ; process's user (u) structure page addr
6778 0000E68A B938000000 <1> mov ecx, (U_SIZE + 3) / 4
6779 0000E68F BE[A44E0100] <1> mov esi, user ; active user (u) structure
6780 0000E694 F3A5 <1> rep movsd
6781 <1> ;
6782 0000E696 8B35[A84E0100] <1> mov esi, [u.usp] ; esp (system stack pointer,
6783 <1> ; points to user registers)
6784 0000E69C 8B0D[A44E0100] <1> mov ecx, [u.sp] ; return address from the system call
6785 <1> ; (for IRET)
6786 <1> ; [u.sp] -> EIP (user)
6787 <1> ; [u.sp+4]-> CS (user)
6788 <1> ; [u.sp+8] -> EFLAGS (user)
6789 <1> ; [u.sp+12] -> ESP (user)
6790 <1> ; [u.sp+16] -> SS (user)
6791 0000E6A2 29F1 <1> sub ecx, esi ; required space for user registers
6792 0000E6A4 83C114 <1> add ecx, 20 ; +5 dwords to return from system call
6793 <1> ; (for IRET)
6794 0000E6A7 C1E902 <1> shr ecx, 2
6795 0000E6AA F3A5 <1> rep movsd
6796 0000E6AC C3 <1> retn
6797 <1>
6798 <1> rswap: ; < swap in, swap from disk >
6799 <1> ; 28/02/2017 (frstor)
6800 <1> ; 15/01/2017
6801 <1> ; 14/01/2017
6802 <1> ; 21/05/2016
6803 <1> ; 03/05/2016
6804 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6805 <1> ; 09/05/2015 - 15/09/2015 (Retro UNIX 386 v1)
6806 <1> ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
6807 <1> ; 'rswap' reads a process whose number is in r1,
6808 <1> ; from disk into core.
6809 <1> ;
6810 <1> ; Retro UNIX 386 v1 modification ->
6811 <1> ; User (u) structure content and the user's register content
6812 <1> ; will be restored from process's/user's UPAGE (a page for
6813 <1> ; saving 'u' structure and user registers for task switching).
6814 <1> ; u.usp - points to kernel stack address which contains
6815 <1> ; user's registers while entering system call.
6816 <1> ; u.sp - points to kernel stack address
6817 <1> ; to return from system call -for IRET-.
6818 <1> ; [u.usp]+32+16 = [u.sp]
6819 <1> ; [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
6820 <1> ; edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
6821 <1> ;
6822 <1> ; RETRO UNIX 8086 v1 modification ->
6823 <1> ; 'swap to disk' is replaced with 'change running segment'
6824 <1> ; according to 8086 cpu (x86 real mode) architecture.
6825 <1> ; pdp-11 was using 64KB uniform memory while IBM PC
6826 <1> ; compatibles was using 1MB segmented memory
6827 <1> ; in 8086/8088 times.
6828 <1> ;
6829 <1> ; INPUTS ->
6830 <1> ; r1 - process number of process to be read in
6831 <1> ; p.break - negative of word count of process
6832 <1> ; p.dska - disk address of the process
6833 <1> ; u.emt - determines handling of emt's
6834 <1> ; u.ilgins - determines handling of illegal instructions
6835 <1> ; OUTPUTS ->
6836 <1> ; 8 = (u.ilgins)
6837 <1> ; 24 = (u.emt)
6838 <1> ; swp - bit 10 is set to indicate read
6839 <1> ; (bit 15=0 when reading is done)
6840 <1> ; swp+2 - disk block address

```

```

6841      <1>      ;   swp+4 - negative word count
6842      <1>      ;   ((swp+6 - address of user structure))
6843      <1>      ;
6844      <1>      ; RETRO UNIX 8086 v1 input/output:
6845      <1>      ;
6846      <1>      ; INPUTS ->
6847      <1>      ;   AL      - new process number (to be swapped in)
6848      <1>      ; OUTPUTS ->
6849      <1>      ;   none
6850      <1>      ;
6851      <1>      ;   ((Modified registers: EAX, ECX, ESI, EDI, ESP))
6852      <1>      ;
6853      <1>      ; Retro UNIX 386 v1 - modification ! 14/05/2015
6854      0000E6AD 89C6      <1>      mov     esi, eax ; process's user (u) structure page addr
6855      0000E6AF B938000000 <1>      mov     ecx, (U_SIZE + 3) / 4
6856      0000E6B4 BF[A44E0100] <1>      mov     edi, user ; active user (u) structure
6857      0000E6B9 F3A5      <1>      rep     movsd
6858      0000E6BB 58        <1>      pop     eax ; 'rswap' return address
6859      <1>      ;
6860      <1>      ;cli
6861      0000E6BC 8B3D[A84E0100] <1>      mov     edi, [u.usp] ; esp (system stack pointer,
6862      <1>      ;           points to user registers)
6863      0000E6C2 89FC      <1>      mov     esp, edi ; 14/01/2017
6864      0000E6C4 8B0D[A44E0100] <1>      mov     ecx, [u.sp] ; return address from the system call
6865      <1>      ; (for IRET)
6866      <1>      ; [u.sp] -> EIP (user)
6867      <1>      ; [u.sp+4]-> CS (user)
6868      <1>      ; [u.sp+8] -> EFLAGS (user)
6869      <1>      ; [u.sp+12] -> ESP (user)
6870      <1>      ; [u.sp+16] -> SS (user)
6871      0000E6CA 29F9      <1>      sub     ecx, edi ; required space for user registers
6872      0000E6CC 83C114    <1>      add     ecx, 20 ; +5 dwords to return from system call
6873      <1>      ; (for IRET)
6874      0000E6CF C1E902    <1>      shr     ecx, 2
6875      0000E6D2 F3A5      <1>      rep     movsd
6876      <1>      ;mov     esp, [u.usp] ; 15/09/2015
6877      <1>      ;sti
6878      <1>      ; 28/02/2017
6879      <1>      ;cmp     byte [multi_tasking], 0 ; Musti tasking mode ?
6880      <1>      ;jna     short rswp_retn
6881      0000E6D4 803D[224F0100]00 <1>      cmp     byte [u.fpsave], 0
6882      0000E6DB 7606      <1>      jna     short rswp_retn
6883      0000E6DD DD25[244F0100] <1>      frstor [u.fpregs] ; restore floating point regs (94 bytes)
6884      <1>      rswp_retn:
6885      0000E6E3 50        <1>      push    eax ; 'rswap' return address
6886      0000E6E4 C3        <1>      retn
6887      <1>
6888      <1>      putlu:
6889      <1>      ; 20/05/2016
6890      <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6891      <1>      ; 10/05/2015 - 12/09/2015 (Retro UNIX 386 v1)
6892      <1>      ; 15/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
6893      <1>      ; 'putlu' is called with a process number in r1 and a pointer
6894      <1>      ; to lowest priority Q (runq+4) in r2. A link is created from
6895      <1>      ; the last process on the queue to process in r1 by putting
6896      <1>      ; the process number in r1 into the last process's link.
6897      <1>      ;
6898      <1>      ; INPUTS ->
6899      <1>      ;   r1 - user process number
6900      <1>      ;   r2 - points to lowest priority queue
6901      <1>      ;   p.dska - disk address of the process
6902      <1>      ;   u.emt - determines handling of emt's
6903      <1>      ;   u.ilgins - determines handling of illegal instructions
6904      <1>      ; OUTPUTS ->
6905      <1>      ;   r3 - process number of last process on the queue upon
6906      <1>      ;   entering putlu
6907      <1>      ;   p.link-1 + r3 - process number in r1
6908      <1>      ;   r2 - points to lowest priority queue
6909      <1>      ;
6910      <1>      ; ((Modified registers: EDX, EBX))
6911      <1>      ;
6912      <1>      ; / r1 = user process no.; r2 points to lowest priority queue
6913      <1>
6914      <1>      ; EBX = r2
6915      <1>      ; EAX = r1 (AL=r1b)
6916      <1>
6917      <1>      ; 20/05/2016
6918      <1>      ; AL = process number (1 to 16) // Retro UNIX 8086, 386 v1 //
6919      <1>      ; (max. 16 processes available for current kernel version)
6920      <1>      ; EBX = run queue address ; 20/05/2016 (TRDOS 386)
6921      <1>      ; which is one of following addresses:
6922      <1>      ;   1) 'runq_event' high priority run queue
6923      <1>      ;   2) 'runq_normal' normal/regular priority run queue
6924      <1>      ;   3) 'runq_background' low priority run queue
6925      <1>
6926      <1>      ;mov     ebx, runq
6927      0000E6E5 0FB613    <1>      movzx    edx, byte [ebx]
6928      0000E6E8 43        <1>      inc     ebx
6929      0000E6E9 20D2      <1>      and     dl, dl
6930      <1>      ; tstb (r2)+ / is queue empty?
6931      0000E6EB 740A      <1>      jz      short putlu_1
6932      <1>      ; beq 1f / yes, branch
6933      0000E6ED 8A13      <1>      mov     dl, [ebx] ; 12/09/2015
6934      <1>      ; movb (r2),r3 / no, save the "last user" process number
6935      <1>      ; / in r3
6936      0000E6EF 8882[E54B0100] <1>      mov     [edx+p.link-1], al
6937      <1>      ; movb r1,p.link-1(r3) / put pointer to user on
6938      <1>      ; / "last users" link
6939      0000E6F5 EB03      <1>      jmp     short putlu_2
6940      <1>      ; br 2f /
6941      <1>      putlu_1: ; 1:
6942      0000E6F7 8843FF    <1>      mov     [ebx-1], al

```



```

6943                                     <1>             ; movb r1,-1(r2) / user is only user;
6944                                     <1>             ; / put process no. at beginning and at end
6945                                     <1> putlu_2: ; 2:
6946 0000E6FA 8803                       <1>         mov     [ebx], al
6947                                     <1>             ; movb r1,(r2) / user process in r1 is now the last entry
6948                                     <1>             ; / on the queue
6949 0000E6FC 88C2                       <1>         mov     dl, al
6950 0000E6FE 88B2[E54B0100]             <1>         mov     [edx+p.link-1], dh ; 0
6951                                     <1>             ; dec r2 / restore r2
6952 0000E704 C3                         <1>         retn
6953                                     <1>             ; rts r0
6954                                     <1> sysver:
6955                                     <1>             ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
6956 0000E705 C705[AC4E0100]0002-       <1>         mov     dword [u.r0], 200h ; AH = major version, AL = minor version
6957 0000E70D 0000                       <1>
6958                                     <1>         jmp     sysret
6959                                     <1> sysreserved1:
6960                                     <1>             ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
6961                                     <1>             ; // name and content will be changed later //
6962 0000E714 C705[AC4E0100]E007-       <1>         mov     dword [u.r0], 2016
6963 0000E71C 0000                       <1>
6964                                     <1>         jmp     sysret
6965                                     <1> syspri: ; change running priority (of the process)
6966                                     <1>             ; 21/05/2016
6967                                     <1>             ; 20/05/2026 - TRDOS 386 (TRDOS v2.0)
6968                                     <1>             ; INPUT ->
6969                                     <1>             ;     BL = priority level
6970                                     <1>             ;     0 = low running priority (running on background)
6971                                     <1>             ;     1 = normal/regular priority (running as regular)
6972                                     <1>             ;     2 = high/event priority (running for event)
6973                                     <1>             ;     >2 = invalid, it will accepted as 2 (event)
6974                                     <1>             ;     0FFh = get/return current running priority only
6975                                     <1>             ; OUTPUT ->
6976                                     <1>             ;     * if current [u.pri] < 2
6977                                     <1>             ;         if BL input < 0FFh ->
6978                                     <1>             ;             [u.pri] is updated as in BL input (0,1,2)
6979                                     <1>             ;         if BL input = 0FFh -> AL = [u.pri] (current)
6980                                     <1>             ;
6981                                     <1>             ;     * if current [u.pri] = 2
6982                                     <1>             ;         if BL input < 0FFh -> cf = 1 & AL = 2
6983                                     <1>             ;         if BL input = 0FFh -> cf = 0 & AL = 2
6984                                     <1>             ;
6985                                     <1>             ; NOTE:
6986                                     <1>             ; If [u.pri] = 2, it can not be changed to 1 or 0;
6987                                     <1>             ; because, run queue of the running process is unspecified
6988                                     <1>             ; at this stage. Process might be started by a timer event
6989                                     <1>             ; or priority might be changed to high by previous
6990                                     <1>             ; 'syspri' system call. In both cases, the process is in
6991                                     <1>             ; 'runq_normal' or 'runq_background' queue.
6992                                     <1>             ; As result of this fact, when the [u.quant] time quantum
6993                                     <1>             ; of the process is elapsed or 'sysrele' system call is
6994                                     <1>             ; instructed by the process, 'tswap' ('tswitch') procedure
6995                                     <1>             ; will be called (to 'swap' or 'switch' out the procedure)
6996                                     <1>             ; and it will not call 'putlu' to add the (stopping)
6997                                     <1>             ; process to relevant run queue when [u.pri] = 2.
6998                                     <1>             ; (Otherwise, it would be possible to add process to
6999                                     <1>             ; a run queue while it is already in a run queue, wrongly.)
7000                                     <1>             ;
7001                                     <1>             ; If [u.pri]< 2, 'tswap/tswitch' procedure will call
7002                                     <1>             ; 'putlu' to add process to relevant run queue
7003                                     <1>             ; according to [u.pri] value. ('runq_normal' for 1,
7004                                     <1>             ; 'runq_background' for 0).
7005                                     <1>             ;
7006                                     <1>             ; If BL input >= 2 and < 0FFh while [u.pri] < 2,
7007                                     <1>             ; process will be added to 'runq_normal' queue and
7008                                     <1>             ; [u.pri] will be set to 2. (in 'syspri' system call)
7009                                     <1>             ;
7010                                     <1>
7011 0000E723 29C0                       <1>         sub     eax, eax ; 0
7012 0000E725 A3[104F0100]               <1>         mov     [u.error], eax
7013                                     <1>
7014 0000E72A A0[F14E0100]               <1>         mov     al, [u.pri]
7015 0000E72F A3[AC4E0100]               <1>         mov     [u.r0], eax
7016                                     <1>
7017 0000E734 FEC3                       <1>         inc     bl
7018 0000E736 0F84D3DCFFFF               <1>         jz      sysret ; 0FFh -> 0, get priority level
7019                                     <1>
7020 0000E73C 3C02                       <1>         cmp     al, 2
7021 0000E73E 0F83ABDCFFFF               <1>         jnb     error ; CF = 1 & AL = 2 (& last error = 0)
7022                                     <1>
7023 0000E744 FECB                       <1>         dec     bl
7024 0000E746 80FB02                     <1>         cmp     bl, 2
7025 0000E749 7602                       <1>         jna     short syspri_1
7026 0000E74B B302                       <1>         mov     bl, 2
7027                                     <1> syspri_1:
7028 0000E74D 881D[F14E0100]             <1>         mov     [u.pri], bl
7029 0000E753 80FB02                     <1>         cmp     bl, 2
7030 0000E756 0F82B3DCFFFF               <1>         jb      sysret
7031                                     <1>
7032                                     <1>             ; here...
7033                                     <1>             ; Priority of current process has been changed to high
7034                                     <1>             ; ('run for event') but current process will be added to
7035                                     <1>             ; 'run as normal' queue. ('run for event' high priority
7036                                     <1>             ; queue is under control of timer -& RTC- interrupt only!)
7037                                     <1>             ;
7038                                     <1>             ; (Otherwise, process can fall into black hole!
7039                                     <1>             ; e.g. if it is not in waiting list and it has not got
7040                                     <1>             ; a timer event and it is not in a run queue!
7041                                     <1>             ; Because, when [u.pri] is 2, 'tswap/tswitch' will not
7042                                     <1>             ; add the stopping process to a run queue.)

```

7043		<1>	
7044	0000E75C A0[FB4E0100]	<1>	mov al, [u.uno]
7045	0000E761 BB[9A4E0100]	<1>	mov ebx, runq_normal ; normal priority !
7046		<1>	; [u.pri] is set to high
7047		<1>	; but 'runq_event' queue is set
7048		<1>	; only by the kernel's timer
7049		<1>	; event function (timer interrupt).
7050	0000E766 E87AFFFFFF	<1>	call putlu
7051	0000E76B E99FDCFFFF	<1>	jmp sysret
7052		<1>	
7053		<1>	cpass: ; / get next character from user area of core and put it in AL (r1)
7054		<1>	; 02/05/2016 - TRDOS 386 (TRDOS v2.0)
7055		<1>	; 19/05/2015 - 18/10/2015 (Retro UNIX 386 v1)
7056		<1>	; 14/08/2013 - 20/09/2013 (Retro UNIX 8086 v1)
7057		<1>	; INPUTS ->
7058		<1>	; [u.base] = virtual address in user area
7059		<1>	; [u.count] = byte count (max.)
7060		<1>	; [u.pcount] = byte count in page (0 = reset)
7061		<1>	; OUTPUTS ->
7062		<1>	; AL = the character which is pointed by [u.base]
7063		<1>	; zf = 1 -> transfer count has been completed
7064		<1>	;
7065		<1>	; ((Modified registers: EAX, EDX, ECX))
7066		<1>	;
7067	0000E770 833D[D04E0100]00	<1>	cmp dword [u.count], 0 ; have all the characters been transferred
7068		<1>	; i.e., u.count, # of chars. left
7069	0000E777 763F	<1>	jna short cpass_3 ; to be transferred = 0?) yes, branch
7070	0000E779 FF0D[D04E0100]	<1>	dec dword [u.count] ; no, decrement u.count
7071		<1>	; 19/05/2015
7072		<1>	;(Retro UNIX 386 v1 - translation from user's virtual address
7073		<1>	to physical address
7074	0000E77F 66833D[0C4F0100]00	<1>	cmp word [u.pcount], 0 ; byte count in page = 0 (initial value)
7075		<1>	; 1-4095 --> use previous physical base address
7076		<1>	; in [u.pbase]
7077	0000E787 770E	<1>	ja short cpass_1
7078	0000E789 833D[044F0100]00	<1>	cmp dword [u.ppgdir], 0 ; is the caller os kernel
7079	0000E790 7427	<1>	je short cpass_k ; (sysexec, '/etc/init') ? (MainProg)
7080	0000E792 E849FDCFFFF	<1>	call trans_addr_r
7081		<1>	cpass_1:
7082	0000E797 66FF0D[0C4F0100]	<1>	dec word [u.pcount]
7083		<1>	cpass_2:
7084	0000E79E 8B15[084F0100]	<1>	mov edx, [u.pbase]
7085	0000E7A4 8A02	<1>	mov al, [edx] ; take the character pointed to
7086		<1>	; by u.base and put it in r1
7087	0000E7A6 FF05[D44E0100]	<1>	inc dword [u.nread] ; increment no. of bytes transferred
7088	0000E7AC FF05[CC4E0100]	<1>	inc dword [u.base] ; increment the buffer address to point to the
7089		<1>	; next byte
7090	0000E7B2 FF05[084F0100]	<1>	inc dword [u.pbase]
7091		<1>	cpass_3:
7092	0000E7B8 C3	<1>	retn
7093		<1>	cpass_k:
7094		<1>	; 02/07/2015
7095		<1>	; The caller is os kernel
7096		<1>	; (get sysexec arguments from kernel's memory space)
7097	0000E7B9 8B1D[CC4E0100]	<1>	mov ebx, [u.base]
7098	0000E7BF 66C705[0C4F0100]00-	<1>	mov word [u.pcount], PAGE_SIZE ; 4096
7099	0000E7C7 10	<1>	
7099	0000E7C8 891D[084F0100]	<1>	mov [u.pbase], ebx
7100	0000E7CE EBCE	<1>	jmp short cpass_2
7101		<1>	
7102		<1>	transfer_to_user_buffer: ; fast transfer
7103		<1>	; 27/05/2016
7104		<1>	; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
7105		<1>	;
7106		<1>	; INPUT ->
7107		<1>	; ESI = source address in system space
7108		<1>	; EDI = user's buffer address
7109		<1>	; ECX = transfer (byte) count
7110		<1>	; [u.pgdir] = user's page directory
7111		<1>	; OUTPUT ->
7112		<1>	; ECX = actual transfer count
7113		<1>	; cf = 1 -> error
7114		<1>	; [u.count] = remain byte count
7115		<1>	;
7116		<1>	; Modified registers: eax, ecx
7117		<1>	;
7118		<1>	
7119	0000E7D0 21C9	<1>	and ecx, ecx
7120	0000E7D2 743B	<1>	jz short ttub_4
7121		<1>	
7122	0000E7D4 890D[D04E0100]	<1>	mov [u.count], ecx
7123		<1>	
7124	0000E7DA 57	<1>	push edi
7125	0000E7DB 56	<1>	push esi
7126	0000E7DC 53	<1>	push ebx
7127	0000E7DD 52	<1>	push edx
7128	0000E7DE 51	<1>	push ecx
7129		<1>	
7130	000		

```

7144
7145 0000E7FB 29C8
7146 0000E7FD 01CB
7147 0000E7FF F3A4
7148 0000E801 A3[D04E0100]
7149 0000E806 09C0
7150 0000E808 75DD
7151
7152
7153 0000E80A 59
7154
7155 0000E80B 5A
7156 0000E80C 5B
7157 0000E80D 5E
7158 0000E80E 5F
7159
7160 0000E80F C3
7161
7162 0000E810 59
7163 0000E811 2B0D[D04E0100]
7164 0000E817 F9
7165 0000E818 EBF1
7166
7167
7168
7169
7170
7171
7172
7173
7174
7175
7176
7177
7178
7179
7180
7181
7182
7183
7184 0000E81A 21C9
7185
7186 0000E81C 74F1
7187
7188 0000E81E 890D[D04E0100]
7189
7190 0000E824 57
7191 0000E825 56
7192 0000E826 53
7193 0000E827 52
7194 0000E828 51
7195
7196 0000E829 89F3
7197 0000E82B 81C300004000
7198
7199
7200
7201 0000E831 E85E6AFFFF
7202
7203 0000E836 72D8
7204
7205
7206 0000E838 89C6
7207 0000E83A A1[D04E0100]
7208 0000E83F 39C1
7209 0000E841 7602
7210 0000E843 89C1
7211
7212 0000E845 29C8
7213 0000E847 01CB
7214 0000E849 F3A4
7215 0000E84B A3[D04E0100]
7216 0000E850 09C0
7217 0000E852 75DD
7218
7219 0000E854 EBB4
7220
7221
7222
7223
7224
7225
7226
7227
7228
7229
7230
7231
7232
7233
7234
7235
7236
7237
7238
7239
7240
7241
7242
7243
7244
7245

<1> ttub_2:
<1>     sub    eax, ecx
<1>     add    ebx, ecx
<1>     rep    movsb
<1>     mov     [u.count], eax
<1>     or      eax, eax
<1>     jnz     short ttub_1
<1> ttub_retn:
<1> tfub_retn:
<1>     pop     ecx ; transfer count = actual transfer count
<1> ttub_3:
<1>     pop     edx
<1>     pop     ebx
<1>     pop     esi
<1>     pop     edi
<1> ttub_4:
<1>     retn
<1> ttub_5:
<1>     pop     ecx
<1>     sub     ecx, [u.count] ; actual transfer count
<1>     stc
<1>     jmp     short ttub_3
<1>
<1> transfer_from_user_buffer: ; fast transfer
<1>     ; 27/05/2016
<1>     ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
<1>     ;
<1>     ; INPUT ->
<1>     ;     ESI = user's buffer address
<1>     ;     EDI = destination address in system space
<1>     ;     ECX = transfer (byte) count
<1>     ;     [u.pgdir] = user's page directory
<1>     ; OUTPUT ->
<1>     ;     ecx = actual transfer count
<1>     ;     cf = 1 -> error
<1>     ;     [u.count] = remain byte count
<1>     ;
<1>     ; Modified registers: eax, ecx
<1>     ;
<1>
<1>     and     ecx, ecx
<1>     ;jz     short tfub_4
<1>     jz      short ttub_4
<1>
<1>     mov     [u.count], ecx
<1>
<1>     push    edi
<1>     push    esi
<1>     push    ebx
<1>     push    edx
<1>     push    ecx
<1>
<1>     mov     ebx, esi
<1>     add     ebx, CORE ; 27/05/2016
<1> tfub_1:
<1>     ; ebx = virtual (linear) address
<1>     ; [u.pgdir] = user's page directory
<1>     call    get_physical_addr_x ; get physical address
<1>     ;jc     short tfub_5
<1>     jc      short ttub_5
<1>     ; eax = physical address
<1>     ; ecx = remain byte count in page (1-4096)
<1>     mov     esi, eax
<1>     mov     eax, [u.count]
<1>     cmp     ecx, eax
<1>     jna     short tfub_2
<1>     mov     ecx, eax
<1> tfub_2:
<1>     sub     eax, ecx
<1>     add     ebx, ecx
<1>     rep    movsb
<1>     mov     [u.count], eax
<1>     or      eax, eax
<1>     jnz     short tfub_1
<1>
<1>     jmp     short tfub_retn
<1>
<1> ;tfub_retn:
<1> ;     pop     ecx ; transfer count = actual transfer count
<1> ;tfub_3:
<1> ;     pop     edx
<1> ;     pop     ebx
<1> ;     pop     esi
<1> ;     pop     edi
<1> ;tfub_4:
<1> ;     retn
<1> ;tfub_5:
<1> ;     pop     ecx
<1> ;     sub     ecx, [u.count] ; actual transfer count
<1> ;     stc
<1> ;     jmp     short tfub_3
<1>
<1> sysfff: ; <Find First File>
<1>     ; 17/10/2016
<1>     ; 16/10/2016
<1>     ; 15/10/2016 TRDOS 386 (TRDOS v2.0) feature only !
<1>     ;         -derived from TRDOS v1.0, INT_21H.ASM-
<1>     ;         ("loc_INT21h_find_first_file")
<1>     ; TRDOS 8086 (v1.0)
<1>     ;     07/08/2011
<1>     ;     Find First File
<1>     ;
<1>     ; INPUT:

```

```

7246 <1> ; CX= Attributes
7247 <1> ; DS:DX= Pointer to filename
7248 <1> ; MSDOS OUTPUT:
7249 <1> ; DTA: (Default address: PSP offset 80h)
7250 <1> ; Offset Description
7251 <1> ; 0 Reserved for use find next file
7252 <1> ; 21 Attribute of file found
7253 <1> ; 22 Time stamp of file
7254 <1> ; 24 Date stamp of file
7255 <1> ; 26 File size in bytes
7256 <1> ; 30 Filename and extension (zero terminated)
7257 <1> ; If cf = 1:
7258 <1> ; Error Codes: (in AX)
7259 <1> ; 2 - File not found
7260 <1> ; 18 - No more files
7261 <1> ;
7262 <1> ; TRDOS 386 (v2.0)
7263 <1> ; 15/10/2016
7264 <1> ;
7265 <1> ; INPUT ->
7266 <1> ; CL = File attributes
7267 <1> ; bit 0 (1) - Read only file (R)
7268 <1> ; bit 1 (1) - Hidden file (H)
7269 <1> ; bit 2 (1) - System file (R)
7270 <1> ; bit 3 (1) - Volume label/name (V)
7271 <1> ; bit 4 (1) - Subdirectory (D)
7272 <1> ; bit 5 (1) - File has been archived (A)
7273 <1> ; CH = 0 -> Return basic parameters (24 bytes)
7274 <1> ; CH > 0 -> Return FindFile structure/table (128 bytes)
7275 <1> ; EBX = Pointer to filename (ASCIIIZ) -path-
7276 <1> ; EDX = File parameters buffer address
7277 <1> ; (buffer size = 24 bytes if CH input = 0)
7278 <1> ; (buffer size = 128 bytes if CH input > 0)
7279 <1> ;
7280 <1> ; OUTPUT ->
7281 <1> ; EAX = 0 if CH input > 0
7282 <1> ; EAX = First cluster number of file if CH input = 0
7283 <1> ; EDX = File parameters table/structure address
7284 <1> ; Basic Parameters:
7285 <1> ; Offset Description
7286 <1> ; -----
7287 <1> ; 0 File Attributes
7288 <1> ; 1 Ambiguous filename chars are used sign
7289 <1> ; (0 = filename fits exactly with request)
7290 <1> ; (>0 = ambiguous filename chars are used)
7291 <1> ; 2 Time stamp of file
7292 <1> ; 4 Date stamp of file
7293 <1> ; 6 File size in bytes
7294 <1> ; 10 Short Filename (ASCIIIZ, max. 13 bytes)
7295 <1> ; 23 Longname Length (1-255) if existing
7296 <1> ;
7297 <1> ; cf = 1 -> Error code in AL
7298 <1> ;
7299 <1> ; Modified Registers: EAX (at the return of system call)
7300 <1> ;
7301 <1> ; TR-DOS FindFile (FFF) Structure (128 bytes):
7302 <1> ; 09/10/2011 (DIR.ASM) - 10/02/2016 (trdoskx.s)
7303 <1> ;
7304 <1> ; Offset Parameter Size
7305 <1> ; -----
7306 <1> ; 0 FindFile_Drv 1 byte
7307 <1> ; 1 FindFile_Directory 65 bytes
7308 <1> ; 66 FindFile_Name 13 bytes
7309 <1> ; 79 FindFile_LongNameEntryLength 1 byte
7310 <1> ; Above 80 bytes form
7311 <1> ; TR-DOS Source/Destination File FullName Format/Structure
7312 <1> ; 80 FindFile_AttributesMask 1 word
7313 <1> ; 82 FindFile_DirEntry 32 bytes (*)
7314 <1> ; 114 FindFile_DirFirstCluster 1 double word
7315 <1> ; 118 FindFile_DirCluster 1 double word
7316 <1> ; 122 FindFile_DirEntryNumber 1 word
7317 <1> ; 124 FindFile_MatchCounter 1 word
7318 <1> ; 126 FindFile_Reserved 1 word
7319 <1> ; (*) MS-DOS, FAT 12-16-32 classic directory entry (32 bytes)
7320 <1> ;
7321 <1> ; mov [u.namep], ebx
7322 <1> ; 16/10/2016
7323 0000E856 8915[D4450100] <1> mov [FFF_UBuffer], edx
7324 0000E85C 66890D[D9450100] <1> mov [FFF_Attrib], cx ; [FFF_RType] = ch
7325 <1> ; Attributes in CL, return data type in CH
7326 0000E863 89DE <1> mov esi, ebx
7327 <1> ; file name is forced, change directory as temporary
7328 <1> ; mov ax, 1
7329 <1> ; mov [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
7330 <1> ; call set_working_path
7331 0000E865 E8720D0000 <1> call set_working_path_x ; 17/10/2016
7332 0000E86A 731D <1> jnc short sysfff_0
7333 <1>
7334 0000E86C 21C0 <1> and eax, eax ; 0 -> Bad Path!
7335 0000E86E 7505 <1> jnz short sysfff_err
7336 <1>
7337 <1> ; eax = 0
7338 0000E870 B80C000000 <1> mov eax, ERR_DIR_NOT_FOUND ; Directory not found !
7339 <1> sysfff_err:
7340 0000E875 A3[AC4E0100] <1> mov [u.r0], eax
7341 0000E87A A3[104F0100] <1> mov [u.error], eax
7342 0000E87F E82D0E0000 <1> call reset_working_path
7343 0000E884 E966DBFFFF <1> jmp error
7344 <1>
7345 <1> sysfff_0:
7346 <1> ; sub ah, ah ; ah = 0
7347 0000E889 8A0424 <1> mov al, [esp]

```


7348	0000E88C	08C0	<1>	or	al, al
7349	0000E88E	7412	<1>	jz	short sysfff_2
7350	0000E890	B410	<1>	mov	ah, 10h
7351	0000E892	A808	<1>	test	al, 08h
7352	0000E894	7503	<1>	jnz	short sysfff_1
7353	0000E896	80CC08	<1>	or	ah, 08h
7354			<1>	sysfff_1:	
7355	0000E899	2410	<1>	and	al, 10h ; Directory
7356	0000E89B	7405	<1>	jz	short sysfff_2
7357	0000E89D	80E408	<1>	and	ah, 08h
7358	0000E8A0	30C0	<1>	xor	al, al ; When a directory is searched,
7359			<1>		; filename will be returned even if
7360			<1>		; it is not a directory!
7361			<1>		; Because: (in order to prevent
7362			<1>		; creating a dir with existing file name)
7363			<1>		; Dir and file names must not be same!
7364			<1>		; (return attribute must be checked)
7365			<1>	sysfff_2:	
7366			<1>		; AX = Attributes mask
7367			<1>		; AL = AND mask (result must be equal to AL)
7368			<1>		; AH = Negative AND mask (result must be ZERO)
7369			<1>		; ESI = FindFile_Name address
7370			<1>		
7371	0000E8A2	E8A896FFFF	<1>	call	find_first_file
7372	0000E8A7	72CC	<1>	jc	short sysfff_err ; eax = 2 (File not found !)
7373			<1>		
7374			<1>		; ESI = Directory Entry (FindFile_DirEntry) Location
7375			<1>		; EDI = Directory Buffer Directory Entry Location
7376			<1>		; EAX = File Size
7377			<1>		; BL = Attributes of The File/Directory
7378			<1>		; BH = Long Name Yes/No Status (>0 is YES)
7379			<1>		; DX > 0 : Ambiguous filename chars are used
7380			<1>		
7381			<1>	sysfff_3:	
7382			<1>		; 16/10/2016
7383	0000E8A9	668B0D[D9450100]	<1>	mov	cx, [FFF_Attrib]
7384			<1>		; Attribs in CL, return data type in CH
7385			<1>		
7386			<1>	;or	cl, cl
7387			<1>	;jz	short sysfff_4 ; 0 = No filter
7388	0000E8B0	80F1FF	<1>	xor	cl, 0FFh
7389	0000E8B3	20D9	<1>	and	cl, bl
7390	0000E8B5	7409	<1>	jz	short sysfff_4
7391			<1>		
7392			<1>	;mov	eax, 2 ; 'file not found !' error
7393			<1>	;jmp	short sysfff_err_1
7394			<1>		
7395			<1>		; 16/10/2016
7396	0000E8B7	E84297FFFF	<1>	call	find_next_file
7397	0000E8BC	72B7	<1>	jc	short sysfff_err ; eax = 12 (no more files !)
7398	0000E8BE	EBE9	<1>	jmp	short sysfff_3
7399			<1>		
7400			<1>	sysfff_4:	
7401	0000E8C0	20ED	<1>	and	ch, ch ; [FFF_RType]
7402	0000E8C2	7412	<1>	jz	short sysfff_5
7403	0000E8C4	B980000000	<1>	mov	ecx, 128 ; ; transfer length
7404	0000E8C9	880D[D8450100]	<1>	mov	[FFF_Valid], cl
7405			<1>	sysfnf_11:	
7406	0000E8CF	BE[8A420100]	<1>	mov	esi, FindFile_Drv
7407	0000E8D4	EB44	<1>	jmp	short sysfff_6
7408			<1>	sysfff_5:	
7409			<1>	;mov	esi, FindFile_DirEntry
7410	0000E8D6	B918000000	<1>	mov	ecx, 24 ; transfer length
7411	0000E8DB	880D[D8450100]	<1>	mov	[FFF_Valid], cl
7412			<1>	sysfnf_12:	
7413	0000E8E1	BF[944A0100]	<1>	mov	edi, DTA ; FFF data transfer address
7414			<1>	;mov	al, [esi+DirEntry_Attr] ; 11
7415	0000E8E6	88D8	<1>	mov	al, bl ; File/Dir Attributes
7416	0000E8E8	887F17	<1>	mov	[edi+23], bh ; Longname length (0= none)
7417	0000E8EB	AA	<1>	stosb	
7418	0000E8EC	88D0	<1>	mov	al, dl ; DL is for '?'
7419	0000E8EE	00F0	<1>	add	al, dh ; DH is for '*'
7420			<1>		; AL > 0 if ambiguous file name wildcards are used
7421	0000E8F0	AA	<1>	stosb	
7422	0000E8F1	8B4616	<1>	mov	eax, [esi+

```

7450      <1>      ; Find First File
7451      <1>      ; INPUT:
7452      <1>      ; none
7453      <1>      ; MSDOS OUTPUT:
7454      <1>      ; DTA: (Default address: PSP offset 80h)
7455      <1>      ; Offset Description
7456      <1>      ; 0 Reserved for use find next file
7457      <1>      ; 21 Attribute of file found
7458      <1>      ; 22 Time stamp of file
7459      <1>      ; 24 Date stamp of file
7460      <1>      ; 26 File size in bytes
7461      <1>      ; 30 Filename and extension (zero terminated)
7462      <1>      ; If cf = 1:
7463      <1>      ; Error Codes: (in AX)
7464      <1>      ; 18 - No more files
7465      <1>      ;
7466      <1>      ; TRDOS 386 (v2.0)
7467      <1>      ; 16/10/2016
7468      <1>      ;
7469      <1>      ; INPUT ->
7470      <1>      ; none
7471      <1>      ; OUTPUT ->
7472      <1>      ; EAX = 0 if CH input of 'Find First File' > 0
7473      <1>      ; EAX = First cluster number of file
7474      <1>      ; if CH input of 'Find First File' = 0
7475      <1>      ; EDX = File parameters table/structure address
7476      <1>      ;
7477      <1>      ; cf = 1 -> Error code in AL
7478      <1>      ;
7479      <1>      ; Modified Registers: EAX (at the return of system call)
7480      <1>      ;
7481      <1>      ; Note: If byte [FFF_Valid] = 0
7482      <1>      ; 'sysfnf' will return with 'no more files' error.
7483      <1>      ; If byte [FFF_Valid] = 24
7484      <1>      ; 'sysfnf' will return with 32 bytes basic parameters
7485      <1>      ; at the address which is in EDX.
7486      <1>      ; If byte [FFF_Valid] = 128
7487      <1>      ; 'sysfnf' will return with 128 bytes Find File
7488      <1>      ; Structure/Table at the address which is in EDX.
7489      <1>      ;
7490      <1>      ;
7491      0000E935 803D[D8450100]00 <1>      cmp byte [FFF_Valid], 0
7492      0000E93C 7714 <1>      ja short stsfnf_0
7493      <1>      ; 'no more files !' error
7494      0000E93E B80C000000 <1>      mov eax, ERR_NO_MORE_FILES ; 12
7495      0000E943 A3[AC4E0100] <1>      mov [u.r0], eax
7496      0000E948 A3[104F0100] <1>      mov [u.error], eax
7497      0000E94D E99DDAFFFF <1>      jmp error
7498      <1>      stsfnf_0:
7499      <1>      ;cmp byte [FFF_Valid], 128
7500      <1>      ;je short stsfnf_1
7501      <1>      ;cmp byte [FFF_Valid], 24
7502      <1>      ;je short stsfnf_1
7503      <1>      ;mov [FFF_Valid], 24 ; Default
7504      <1>      stsfnf_1:
7505      0000E952 0FB61D[E6380100] <1>      movzx ebx, byte [Current_Drv]
7506      0000E959 66891D[DE450100] <1>      mov [SWP_DRV], bx
7507      0000E960 8A15[8A420100] <1>      mov dl, [FindFile_Drv]
7508      0000E966 38DA <1>      cmp dl, bl
7509      0000E968 750B <1>      jne short stsfnf_2
7510      0000E96A 86FB <1>      xchg bh, bl
7511      0000E96C BE00010900 <1>      mov esi, Logical_DOSDisks
7512      0000E971 01DE <1>      add esi, ebx
7513      0000E973 EB0D <1>      jmp short sysfnf_3
7514      <1>
7515      <1>      stsfnf_2:
7516      0000E975 FE05[DF450100] <1>      inc byte [SWP_DRV_chg]
7517      <1>
7518      0000E97B E83D82FFFF <1>      call change_current_drive
7519      0000E980 7245 <1>      jc short sysfnf_err_1 ; read error !
7520      <1>      ; (do not stop, because
7521      <1>      ; we don't have a
7522      <1>      ; 'no more files'
7523      <1>      ; -file not found- error,
7524      <1>      ; next sysfnf system call
7525      <1>      ; may solve the problem,
7526      <1>      ; after re-placing the disk)
7527      <1>      sysfnf_3:
7528      0000E982 A1[00430100] <1>      mov eax, [FindFile_DirCluster]
7529      0000E987 21C0 <1>      and eax, eax
7530      0000E989 7550 <1>      jnz short sysfnf_6
7531      <1>
7532      0000E98B 803D[E5380100]02 <1>      cmp byte [Current_FATType], 2
7533      0000E992 772C <1>      ja short sysfnf_err_0 ; invalid, we needed to stop !?
7534      0000E994 803D[E5380100]01 <1>      cmp byte [Current_FATType], 1
7535      0000E99B 7223 <1>      jb short sysfnf_err_0 ; invalid, we needed to stop !?
7536      <1>
7537      0000E99D 3805[10410100] <1>      cmp byte [DirBuff_ValidData], al ; 0
7538      0000E9A3 7608 <1>      jna short sysfnf_4
7539      <1>
7540      0000E9A5 3B05[15410100] <1>      cmp eax, [DirBuff_Cluster] ; 0 ?
7541      0000E9AB 745E <1>      je short sysfnf_9
7542      <1>
7543      <1>      ;cmp byte [Current_Dir_Level], 0
7544      <1>      ;ja short sysfnf_4
7545      <1>      ;jna short sysfnf_9
7546      <1>
7547      <1>      sysfnf_4:
7548      0000E9AD FE05[DF450100] <1>      inc byte [SWP_DRV_chg]
7549      0000E9B3 E820D0FFFF <1>      call load_FAT_root_directory
7550      0000E9B8 7351 <1>      jnc short sysfnf_9
7551      <1>      ; eax = error code (17, 'drv not ready or read error')

```

```

7552 0000E9BA EB0B      <1>      jmp      short sysfnf_err_1 ; read error ! (no FNF stop)
7553                    <1>                    ; (if you want, try again,
7554                    <1>                    ; after re-placing the disk)
7555                    <1> sysfnf_5:
7556 0000E9BC 3C0C      <1>      cmp      al, 12 ; 'no more files' error
7557 0000E9BE 7507      <1>      jne      short sysfnf_err_1 ; (no FNF stop -sysfnf will try
7558                    <1>                    ; to read the directory again,
7559                    <1>                    ; if the user calls sysfnf
7560                    <1>                    ; just after this error return-)
7561                    <1>                    ; (FNF stop -sysfnf will not try
7562                    <1>                    ; to read the directory again-)
7563                    <1>
7564                    <1> sysfnf_err_0:
7565 0000E9C0 C605[D8450100]00 <1>      mov      byte [FFF_Valid], 0 ; FNF stop sign
7566                    <1> sysfnf_err_1:
7567 0000E9C7 A3[AC4E0100] <1>      mov      [u.r0], eax
7568 0000E9CC A3[104F0100] <1>      mov      [u.error], eax
7569 0000E9D1 E8DB0C0000 <1>      call     reset_working_path
7570 0000E9D6 E914DAFFFF <1>      jmp      error
7571                    <1>
7572                    <1> sysfnf_6:
7573 0000E9DB 803D[10410100]00 <1>      cmp      byte [DirBuff_ValidData], 0
7574 0000E9E2 7608      <1>      jna      short sysfnf_7
7575                    <1>
7576 0000E9E4 3B05[15410100] <1>      cmp      eax, [DirBuff_Cluster]
7577 0000E9EA 741F      <1>      je      short sysfnf_9
7578                    <1>
7579                    <1> sysfnf_7:
7580 0000E9EC FE05[DF450100] <1>      inc      byte [SWP_DRV_chg]
7581 0000E9F2 803D[E5380100]01 <1>      cmp      byte [Current_FATType], 1
7582 0000E9F9 7309      <1>      jnb      short sysfnf_8
7583                    <1>
7584                    <1> ; Singlix (TRFS) File System
7585                    <1> ; (access via compatibility buffer)
7586 0000E9FB E8A0D0FFFF <1>      call     load_FS_sub_directory
7587 0000EA00 7309      <1>      jnc      short sysfnf_9
7588                    <1>
7589 0000EA02 EBC3      <1>      jmp      short sysfnf_err_1 ; read error (no FNF stop)
7590                    <1>
7591                    <1> sysfnf_8:
7592 0000EA04 E85AD0FFFF <1>      call     load_FAT_sub_directory
7593 0000EA09 72BC      <1>      jc      short sysfnf_err_1 ; read error (no FNF stop)
7594                    <1>
7595                    <1> sysfnf_9:
7596 0000EA0B E8EE95FFFF <1>      call     find_next_file
7597 0000EA10 72AA      <1>      jc      short sysfnf_5
7598                    <1>
7599 0000EA12 A0[D9450100] <1>      mov      al, [FFF_Attrib]
7600                    <1>      ;or      al, al
7601                    <1>      ;jz      short sysfnf_10 ; 0 = No filter
7602 0000EA17 34FF      <1>      xor      al, 0FFh
7603 0000EA19 20D8      <1>      and      al, bl
7604 0000EA1B 75EE      <1>      jnz      short sysfnf_9 ; search for next file until
7605                    <1>                    ; an error return from
7606                    <1>                    ; find_next_file procedure
7607                    <1> sysfnf_10:
7608 0000EA1D 0FB60D[D8450100] <1>      movzx     ecx, byte [FFF_Valid]
7609 0000EA24 80F980      <1>      cmp      cl, 128 ; complete FindFile structure/table
7610 0000EA27 0F84A2FEFFFF <1>      je      sysfnf_11
7611                    <1>      ;cmp      cl, 24 ; basic parameters
7612                    <1>      ;je      sysfnf_12
7613 0000EA2D E9AFFEFFFF <1>      jmp      sysfnf_12
7614                    <1>
7615                    <1> writei:
7616                    <1> ; 26/10/2016
7617                    <1> ; 25/10/2016
7618                    <1> ; 23/10/2016
7619                    <1> ; 22/10/2016
7620                    <1> ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)
7621                    <1> ; 19/05/2015 - 20/05/2015 (Retro UNIX 386 v1)
7622                    <1> ; 12/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
7623                    <1> ;
7624                    <1> ; Write data to file with first cluster number in EAX
7625                    <1> ;
7626                    <1> ; INPUTS ->
7627                    <1> ; EAX - First cluster number of the file
7628                    <1> ; EBX - File number (Open file index number)
7629                    <1> ; u.count - byte count to be written
7630                    <1> ; u.base - points to user buffer
7631                    <1> ; u.fofp - points to dword with current file offset
7632                    <1> ; i.size - file size
7633                    <1> ; cdev - logical dos drive number of the file
7634                    <1> ; OUTPUTS ->
7635                    <1> ; u.count - cleared
7636                    <1> ; u.nread - accumulates total bytes passed back
7637                    <1> ; i.size - new file size (if file byte offset overs file size)
7638                    <1> ; u.fofp - points to u.off (with new offset value)
7639                    <1> ;
7640                    <1> ; (Retro UNIX Prototype : 11/11/2012 - 18/11/2012, UNIXCOPY.ASM)
7641                    <1> ; ((Modified registers: eax, edx, ebx, ecx, esi, edi, ebp))
7642                    <1>
7643 0000EA32 31C9      <1>      xor      ecx, ecx
7644 0000EA34 890D[D44E0100] <1>      mov      [u.nread], ecx ; 0
7645 0000EA3A 66890D[0C4F0100] <1>      mov      [u.pcount], cx ; 19/05/2015
7646 0000EA41 390D[D04E0100] <1>      cmp      [u.count], ecx
7647 0000EA47 7701      <1>      ja      short writei_1
7648 0000EA49 C3      <1>      retn
7649                    <1> writei_1:
7650 0000EA4A 881D[98450100] <1>      mov      [writei.ofn], bl ; Open file number
7651 0000EA50 880D[D3450100] <1>      mov      [setfmod], cl ; 0 ; reset 'update lm date&time' sign
7652                    <1> dskw_0:
7653                    <1> ; 26/10/2016

```

```

7654 <1> ; 22/10/2016, 23/10/2016, 25/10/2016
7655 <1> ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)
7656 <1> ; 31/05/2015 - 25/07/2015 (Retro UNIX 386 v1)
7657 <1> ; 26/04/2013 - 20/09/2013 (Retro UNIX 8086 v1)
7658 <1> ;
7659 <1> ; 01/08/2013 (mkdir_w check)
7660 0000EA56 E8D7000000 <1> call mget_w
7661 <1> ; eax = sector/block number
7662 <1>
7663 0000EA5B 8B1D[BC4E0100] <1> mov ebx, [u.fofp]
7664 0000EA61 8B13 <1> mov edx, [ebx]
7665 0000EA63 81E2FF010000 <1> and edx, 1FFh ; / test the lower 9 bits of the file offset
7666 0000EA69 750C <1> jnz short dskw_1 ; / if its non-zero, branch
7667 <1> ; if zero, file offset = 0,
7668 <1> ; / 512, 1024,...(i.e., start of new block)
7669 0000EA6B 813D[D04E0100]0002- <1> cmp dword [u.count], 512
7670 0000EA73 0000 <1>
7671 <1> ; / if zero, is there enough data to fill
7672 0000EA75 7337 <1> jnb short dskw_2 ; / an entire block? (i.e., no. of
7673 <1> ; / bytes to be written greater than 512.?
7674 <1> ; / Yes, branch. Don't have to read block
7675 <1> dskw_1: ; in as no past info. is to be saved
7676 <1> ; (the entire block will be overwritten).
7677 <1> ; 23/10/2016
7678 0000EA77 BB[DC520100] <1> mov ebx, writei_buffer
7679 <1> ; esi = logical dos drive description table address
7680 <1> ; eax = sector number
7681 <1> ; ebx = buffer address (in kernel's memory space)
7682 <1> ; ecx = sector count
7683 0000EA7C B901000000 <1> mov ecx, 1
7684 0000EA81 E851070000 <1> call disk_read
7685 <1> ;call dskrd ; / no, must retain old info..
7686 <1> ; / Hence, read block 'r1' into an I/O buffer
7687 0000EA86 7326 <1> jnc short dskw_2
7688 <1>
7689 <1> ; disk read error
7690 0000EA88 B811000000 <1> mov eax, 17 ; drive not ready or READ ERROR !
7691 <1> dskw_err: ; jump from disk write error
7692 0000EA8D A3[AC4E0100] <1> mov [u.r0], eax
7693 0000EA92 A3[104F0100] <1> mov [u.error], eax
7694 <1>
7695 0000EA97 803D[D3450100]00 <1> cmp byte [setfmod], 0
7696 0000EA9E 0F864BD9FFFF <1> jna error
7697 <1>
7698 0000EAA4 E8AF030000 <1> call update_file_lmdt ; update last modif. date&time of the file
7699 <1> ;mov byte [setfmod], 0
7700 <1>
7701 0000EAA9 E941D9FFFF <1> jmp error
7702 <1>
7703 <1> dskw_2: ; 3:
7704 <1> ; 23/10/2016
7705 0000EAAE C605[74450100]01 <1> mov byte [writei.valid], 1 ; writei buffer contains valid data
7706 0000EAB5 56 <1> push esi ; logical dos drive description table address
7707 <1> ; EAX (r1) = block/sector number
7708 <1> ;call wslot
7709 <1> ; jsr r0,wslot / set write and inhibit bits in I/O queue,
7710 <1> ; / proc. status=0, r5 points to 1st word of data
7711 0000EAB6 803D[0E4F0100]00 <1> cmp byte [u.kcall], 0
7712 0000EABD 770F <1> ja short dskw_4 ; zf=0 -> the caller is 'mkdir'
7713 <1> ;
7714 0000EABF 66833D[0C4F0100]00 <1> cmp word [u.pcount], 0
7715 0000EAC7 7705 <1> ja short dskw_4
7716 <1> dskw_3:
7717 <1> ; [u.base] = virtual address to transfer (as source address)
7718 0000EAC9 E812FAFFFF <1> call trans_addr_r ; translate virtual address to physical (r)
7719 <1> dskw_4:
7720 0000EACE BB[DC520100] <1> mov ebx, writei_buffer
7721 <1> ; EBX (r5) = system (I/O) buffer address
7722 0000EAD3 E874FAFFFF <1> call sioreg
7723 <1> ; ESI = file (user data) offset
7724 <1> ; EDI = sector (I/O) buffer offset
7725 <1> ; ECX = byte count
7726 <1> ;
7727 0000EAD8 F3A4 <1> rep movsb
7728 <1> ; 25/07/2015
7729 <1> ; eax = remain bytes in buffer
7730 <1> ; (check if remain bytes in the buffer > [u.pcount])
7731 0000EADA 09C0 <1> or eax, eax
7732 0000EADC 75EB <1> jnz short dskw_3 ; (page end before system buffer end!)
7733 <1>
7734 <1> ; 23/10/2016
7735 0000EADE B101 <1> mov cl, 1
7736 0000EAE0 5E <1> pop esi
7737 0000EAE1 A1[78450100] <1> mov eax, [writei.sector]
7738 <1> ; esi = logical dos drive description table address
7739 <1> ; eax = sector number
7740 <1> ; ebx = writei buffer address
7741 <1> ; ecx = sector count
7742 0000EAE6 E8DD060000 <1> call disk_write ; / yes, write the block
7743 0000EAEB 7307 <1> jnc short dskw_5
7744 <1>
7745 0000EAED B812000000 <1> mov eax, 18 ; drive not ready or WRITE ERROR !
7746 0000EAF2 EB99 <1> jmp short dskw_err
7747 <1>
7748 <1> dskw_5:
7749 <1> ; 26/10/2016
7750 0000EAF4 0FB61D[98450100] <1> movzx ebx, byte [writei.ofn] ; open file number
7751 0000EAFB C0E302 <1> shl bl, 2 ; *4
7752 0000EAFE 8B83[68490100] <1> mov eax, [ebx+OF_POINTER]
7753 0000EB04 3B83[90490100] <1> cmp eax, [ebx+OF_SIZE]
7754 0000EB0A 7606 <1> jna short dskw_6

```



```

7755 0000EB0C 8983[90490100] <1>      mov     [ebx+OF_SIZE], eax
7756 <1> dskw_6:
7757 <1>      ;shr     bl, 2
7758 0000EB12 833D[D04E0100]00 <1>      cmp     dword [u.count], 0 ; / any more data to write?
7759 0000EB19 760A <1>      jna     short dskw_7
7760 0000EB1B A1[88450100] <1>      mov     eax, [writei.fclust]
7761 0000EB20 E931FFFFFF <1>      jmp     dskw_0 ; / yes, branch
7762 <1> dskw_7:
7763 <1>      ; update last modif. date&time of the file
7764 <1>      ; (also updates file size as OF_SIZE)
7765 0000EB25 E82E030000 <1>      call    update_file_lmdt
7766 <1>      ;mov     byte [setfmod], 0
7767 <1>
7768 <1>      ; 03/08/2013
7769 0000EB2A C605[0E4F0100]00 <1>      mov     byte [u.kcall], 0
7770 <1>      ; 23/10/2016
7771 <1>      ;mov     eax, [writei.fclust]
7772 0000EB31 C3 <1>      retn
7773 <1>
7774 <1> mget_w:
7775 <1>      ; 02/11/2016
7776 <1>      ; 01/11/2016
7777 <1>      ; 23/10/2016, 31/10/2016
7778 <1>      ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)
7779 <1>      ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)
7780 <1>      ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
7781 <1>      ;
7782 <1>      ; Get existing or (allocate) a new disk block for file
7783 <1>      ;
7784 <1>      ; INPUTS ->
7785 <1>      ; [u.fofp] = file offset pointer
7786 <1>      ; [i.size] = file size
7787 <1>      ; [u.count] = byte count
7788 <1>      ; EAX = First cluster
7789 <1>      ; [cdev] = Logical dos drive number
7790 <1>      ; [writei.ofn] = File Number
7791 <1>      ; (Open file index, 0 based)
7792 <1>      ; ([u.off] = file offset)
7793 <1>      ; OUTPUTS ->
7794 <1>      ; EAX = logical sector number
7795 <1>      ; ESI = Logical Dos Drive Description Table address
7796 <1>      ;
7797 <1>      ; Modified registers: EDX, EBX, ECX, ESI, EDI, EBP
7798 <1>
7799 0000EB32 8B35[BC4E0100] <1>      mov     esi, [u.fofp]
7800 0000EB38 8B2E <1>      mov     ebp, [esi] ; u.off (or EBX*4+OF_POINTER)
7801 <1>
7802 0000EB3A 29C9 <1>      sub     ecx, ecx
7803 0000EB3C 8A2D[8C4E0100] <1>      mov     ch, [cdev]
7804 <1>
7805 0000EB42 BE00010900 <1>      mov     esi, Logical_DOSDisks
7806 0000EB47 01CE <1>      add     esi, ecx
7807 <1>
7808 <1>      ; 31/10/2016
7809 0000EB49 89C3 <1>      mov     ebx, eax ; First Cluster or FDT address
7810 <1>
7811 0000EB4B 807E0300 <1>      cmp     byte [esi+LD_FATType], 0
7812 0000EB4F 0F86DD010000 <1>      jna     mget_w_14 ; Singlix FS
7813 <1>
7814 0000EB55 0FB74611 <1>      movzx   eax, word [esi+LD_BPB+BytesPerSec]
7815 0000EB59 0FB65613 <1>      movzx   edx, byte [esi+LD_BPB+SecPerClust]
7816 0000EB5D 8815[76450100] <1>      mov     [writei.spc], dl ; sectors per cluster
7817 0000EB63 F7E2 <1>      mul     edx
7818 <1>      ; edx = 0
7819 <1>      ; eax = bytes per cluster (<= 65536)
7820 <1>
7821 <1>      ; 02/11/2016
7822 0000EB65 89C1 <1>      mov     ecx, eax
7823 0000EB67 48 <1>      dec     eax
7824 0000EB68 66A3[7C450100] <1>      mov     [writei.bpc], ax
7825 <1>
7826 0000EB6E 89E8 <1>      mov     eax, ebp
7827 0000EB70 0305[D04E0100] <1>      add     eax, [u.count] ; next file position
7828 0000EB76 3B05[9D4F0100] <1>      cmp     eax, [i.size] ; <= file size ?
7829 0000EB7C 0F86FC000000 <1>      jna     mget_w_4 ; no
7830 <1>
7831 0000EB82 F7F1 <1>      div     ecx
7832 0000EB84 A3[84450100] <1>      mov     [writei.c_index], eax ; cluster index
7833 <1>      ; edx = byte offset in cluster (<= 65535)
7834 <1>      ;mov     [writei.offset], dx
7835 <1>      ;shr     dx, 9 ; / 512
7836 <1>      ;mov     [writei.s_index], dl ; sector index in cluster (0 to spc -1)
7837 <1>
7838 0000EB89 29D2 <1>      sub     edx, edx ; 01/11/2016
7839 0000EB8B 8915[78450100] <1>      mov     [writei.sector], edx ; 0
7840 0000EB91 668915[7E450100] <1>      mov     [writei.offset], dx ; byte offset in cluster
7841 0000EB98 8815[77450100] <1>      mov     [writei.s_index], dl ; sector index in cluster (0 to spc -1)
7842 <1>
7843 0000EB9E 89D8 <1>      mov     eax, ebx ; First Cluster
7844 <1>
7845 <1>      ; is this the 1st mget_w or a next mget_w call ? (by 'writei')
7846 0000EBA0 3815[74450100] <1>      cmp     byte [writei.valid], dl ; 0
7847 0000EBA6 7624 <1>      jna     short mget_w_0
7848 <1>
7849 0000EBA8 8815[74450100] <1>      mov     byte [writei.valid], dl ; 0 ; reset ('writei' will set it)
7850 <1>
7851 0000EBAE 3B05[88450100] <1>      cmp     eax, [writei.fclust]
7852 0000EBB4 7516 <1>      jne     short mget_w_0
7853 <1>
7854 0000EBB6 8A0D[8C4E0100] <1>      mov     cl, [cdev]
7855 0000EBBC 3A0D[75450100] <1>      cmp     cl, [writei.driv]
7856 0000EBC2 7508 <1>      jne     short mget_w_0

```

```

7857 <1> ; [writei.l_clust] & [writei.l_index] are valid,
7858 <1> ; we don't need to get last cluster & last cluster index
7859 0000EBC4 8B0D[94450100] <1> mov ecx, [writei.l_index]
7860 0000EBCA EB64 <1> jmp short mget_w_2
7861 <1> mget_w_0:
7862 0000EBCC A3[88450100] <1> mov [writei.fclust], eax ; first cluster
7863 <1> ; edx = 0
7864 0000EBD1 A3[80450100] <1> mov [writei.cluster], eax ; first cluster ; 01/11/2016
7865 0000EBD6 8915[8C450100] <1> mov [writei.fs_index], edx ; 0 ; curret cluster index
7866 <1>
7867 <1> ; FAT file system (FAT12, FAT16, FAT32)
7868 0000EBDC E897D4FFFF <1> call get_last_cluster
7869 0000EBE1 0F822B010000 <1> jc mget_w_err ; eax = error code
7870 <1>
7871 0000EBE7 A3[90450100] <1> mov [writei.lclust], eax ; last cluster
7872 <1>
7873 0000EBEC 8B0D[B4430100] <1> mov ecx, [glc_index] ; last cluster index
7874 0000EBF2 890D[94450100] <1> mov [writei.l_index], ecx
7875 <1>
7876 0000EBF8 A0[98450100] <1> mov al, [writei.ofn]
7877 0000EBFD FEC0 <1> inc al
7878 0000EBFF A2[D3450100] <1> mov [setfmod], al ; update lm date&time sign
7879 <1>
7880 <1> mget_w_1:
7881 0000EC04 3B0D[84450100] <1> cmp ecx, [writei.c_index] ; last cluster index
7882 0000EC0A 7324 <1> jnb short mget_w_2 ; 01/11/2016
7883 <1>
7884 0000EC0C A1[90450100] <1> mov eax, [writei.lclust]
7885 <1> ; EAX = Last cluster
7886 0000EC11 E870D5FFFF <1> call add_new_cluster
7887 0000EC16 0F82F6000000 <1> jc mget_w_err ; eax = error code
7888 <1> ; edx = 0
7889 0000EC1C A3[90450100] <1> mov [writei.lclust], eax ; (new) last cluster
7890 0000EC21 8B0D[94450100] <1> mov ecx, [writei.l_index]
7891 0000EC27 41 <1> inc ecx ; add 1 to last cluster index
7892 0000EC28 890D[94450100] <1> mov [writei.l_index], ecx ; current last cluster index
7893 <1>
7894 0000EC2E EBD4 <1> jmp short mget_w_1
7895 <1>
7896 <1> mget_w_2:
7897 0000EC30 89E9 <1> mov ecx, ebp
7898 0000EC32 030D[D04E0100] <1> add ecx, [u.count]
7899 0000EC38 890D[9D4F0100] <1> mov [i.size], ecx ; save new file size
7900 <1> ;sub edx, edx ; 0
7901 <1>
7902 0000EC3E A0[8C4E0100] <1> mov al, [cdev]
7903 0000EC43 A2[75450100] <1> mov [writei.driv], al ; physical drive number
7904 <1> ; edx = 0
7905 0000EC48 89E8 <1> mov eax, ebp ; file offset
7906 0000EC4A 0FB70D[7C450100] <1> movzx ecx, word [writei.bpc] ; bytes per cluster - 1
7907 0000EC51 41 <1> inc ecx ; bytes per cluster
7908 0000EC52 F7F1 <1> div ecx
7909 <1> ; edx = byte offset in cluster (<= 65535)
7910 <1> ; eax = cluster index
7911 0000EC54 A3[84450100] <1> mov [writei.c_index], eax
7912 0000EC59 668915[7E450100] <1> mov [writei.offset], dx
7913 0000EC60 66C1EA09 <1> shr dx, 9 ; / 512
7914 0000EC64 8815[77450100] <1> mov [writei.s_index], dl ; sector index in cluster (0 to spc -1)
7915 <1>
7916 <1> mget_w_3:
7917 0000EC6A 3B05[94450100] <1> cmp eax, [writei.l_index] ; last cluster index
7918 0000EC70 752A <1> jne short mget_w_5
7919 <1>
7920 0000EC72 A3[8C450100] <1> mov [writei.fs_index], eax ; cluster index (for next check)
7921 0000EC77 A1[90450100] <1> mov eax, [writei.lclust] ; last cluster
7922 0000EC7C EB60 <1> jmp short mget_w_10
7923 <1>
7924 <1> mget_w_4: ; 02/11/2016
7925 <1> ; eax = next file position
7926 0000EC7E 2B05[D04E0100] <1> sub eax, [u.count] ; current file position
7927 <1> ; edx = 0
7928 <1> ; ecx = bytes per cluster
7929 0000EC84 F7F1 <1> div ecx
7930 0000EC86 A3[84450100] <1> mov [writei.c_index], eax ; cluster index
7931 0000EC8B 668915[7E450100] <1> mov [writei.offset], dx
7932 0000EC92 66C1EA09 <1> shr dx, 9 ; / 512
7933 0000EC96 8815[77450100] <1> mov [writei.s_index], dl ; sector index in cluster (0 to spc -1)
7934 <1>
7935 <1> mget_w_5:
7936 0000EC9C 21C0 <1> and eax, eax ; 0 = First Cluster's index number
7937 0000EC9E 750C <1> jnz short mget_w_6
7938 <1>
7939 0000ECA0 A3[8C450100] <1> mov [writei.fs_index], eax ; cluster index (for next check)
7940 0000ECA5 A1[88450100] <1> mov eax, [writei.fclust] ; first cluster
7941 0000ECAA EB32 <1> jmp short mget_w_10
7942 <1>
7943 <1> mget_w_6:
7944 0000ECAC 3B05[8C450100] <1> cmp eax, [writei.fs_index] ; current cluster index (>0)
7945 0000ECB2 7507 <1> jne short mget_w_7
7946 0000ECB4 A1[80450100] <1> mov eax, [writei.cluster] ; current cluster
7947 0000ECB9 EB3A <1> jmp short mget_w_11
7948 <1>
7949 <1> mget_w_7:
7950 0000ECBB 89C1 <1> mov ecx, eax
7951 0000ECBD 2B0D[8C450100] <1> sub ecx, [writei.fs_index]
7952 0000ECC3 730D <1> jnc short mget_w_8
7953 <1> ; get cluster by index from the first cluster
7954 0000ECC5 A1[88450100] <1> mov eax, [writei.fclust]
7955 0000ECCA 8B0D[84450100] <1> mov ecx, [writei.c_index]
7956 0000ECD0 EB05 <1> jmp short mget_w_9
7957 <1>
7958 <1> mget_w_8:

```

```

7959 0000ECD2 A1[80450100] <1>      mov     eax, [writei.cluster] ; beginning cluster
7960 <1>      ; ecx = cluster sequence number after the beginning cluster
7961 <1>      ; sub     edx, edx ; 0
7962 <1>
7963 <1> mget_w_9:
7964 <1>      ; EAX = Beginning cluster
7965 <1>      ; EDX = Sector index in disk/file section
7966 <1>      ;      (Only for SINGLIX file system!)
7967 <1>      ; ECX = Cluster sequence number after the beginning cluster
7968 <1>      ; ESI = Logical DOS Drive Description Table address
7969 0000ECD7 E8B0D5FFFF <1>      call    get_cluster_by_index
7970 0000ECDC 7234 <1>      jc      short mget_w_err ; error code in EAX
7971 <1>      ; EAX = Cluster number
7972 <1> mget_w_10:
7973 0000ECDE A3[80450100] <1>      mov     [writei.cluster], eax ; FDT number for Singlix File System
7974 <1>
7975 0000ECE3 807E0300 <1>      cmp     byte [esi+LD_FATType], 0
7976 0000ECE7 7638 <1>      jna     short mget_w_13
7977 <1>      ; 01/11/2016
7978 0000ECE9 8B15[84450100] <1>      mov     edx, [writei.c_index]
7979 0000ECE7 8915[8C450100] <1>      mov     [writei.fs_index], edx
7980 <1> mget_w_11:
7981 0000ECF5 83E802 <1>      sub     eax, 2
7982 0000ECF8 0FB615[76450100] <1>      movzx   edx, byte [writei.spc]
7983 0000ECFF F7E2 <1>      mul     edx
7984 <1>
7985 0000ED01 034668 <1>      add     eax, [esi+LD_DATABegin]
7986 0000ED04 8A15[77450100] <1>      mov     dl, [writei.s_index]
7987 0000ED0A 01D0 <1>      add     eax, edx
7988 <1> mget_w_12:
7989 0000ED0C A3[78450100] <1>      mov     [writei.sector], eax
7990 <1>      ; ; buffer validation must be done in writei
7991 <1>      ; ;mov byte [writei.valid], 1
7992 0000ED11 C3 <1>      retn
7993 <1>
7994 <1> mget_w_err:
7995 0000ED12 A3[104F0100] <1>      mov     [u.error], eax
7996 0000ED17 A3[AC4E0100] <1>      mov     [u.r0], eax
7997 0000ED1C E9CED6FFFF <1>      jmp     error
7998 <1>
7999 <1> mget_w_13:
8000 <1>      ; EAX = FDT number (Current Section)
8001 <1>      ; EDX = Sector index from the first section (0,1,2,3,4...)
8002 0000ED21 2B15[8C450100] <1>      sub     edx, [writei.fs_index]
8003 <1>      ; EDX = Sector index from current section
8004 0000ED27 8915[8C450100] <1>      mov     [writei.fs_index], edx
8005 0000ED2D 40 <1>      inc     eax ; the first data sector in FS disk section
8006 0000ED2E 01D0 <1>      add     eax, edx
8007 0000ED30 EBDA <1>      jmp     short mget_w_12
8008 <1>
8009 <1> mget_w_14:
8010 0000ED32 8A4E12 <1>      mov     cl, [esi+LD_FS_BytesPerSec+1]
8011 0000ED35 D0E9 <1>      shr     cl, 1 ; ; 1 for 512 bytes, 4 for 2048 bytes
8012 0000ED37 880D[76450100] <1>      mov     [writei.spc], cl ; sectors per cluster
8013 <1>      ; NOTE: writei bytes per sector value is always 512 !
8014 0000ED3D 66C705[7C450100]00- <1>      mov     word [writei.bpc], 512
8014 0000ED45 02 <1>
8015 <1>
8016 0000ED46 89E9 <1>      mov     ecx, ebp
8017 0000ED48 030D[D04E0100] <1>      add     ecx, [u.count] ; next file position
8018 0000ED4E 3B0D[9D4F0100] <1>      cmp     ecx, [i.size] ; <= file size ?
8019 0000ED54 0F86C8000000 <1>      jna     mget_w_19 ; no
8020 <1>
8021 0000ED5A 29D2 <1>      sub     edx, edx ; 0
8022 0000ED5C 8915[78450100] <1>      mov     [writei.sector], edx ; 0
8023 0000ED62 668915[7E450100] <1>      mov     [writei.offset], dx ; byte offset in cluster
8024 0000ED69 8815[77450100] <1>      mov     [writei.s_index], dl ; sector index in cluster (0 to spc -1)
8025 <1>
8026 0000ED6F C1E909 <1>      shr     ecx, 9 ; 1 cluster = 512 bytes
8027 0000ED72 890D[84450100] <1>      mov     [writei.c_index], ecx ; section/cluster index
8028 <1>
8029 0000ED78 89D8 <1>      mov     eax, ebx ; FDT number (First FDT address)
8030 <1>
8031 <1>      ; is this the 1st mget_w or a next mget_w call ? (by 'writei')
8032 0000ED7A 3815[74450100] <1>      cmp     byte [writei.valid], dl ; 0
8033 0000ED80 7624 <1>      jna     short mget_w_15
8034 <1>
8035 0000ED82 8815[74450100] <1>      mov     byte [writei.valid], dl ; 0 ; reset ('writei' will set it)
8036 <1>
8037 0000ED88 3B05[88450100] <1>      cmp     eax, [writei.fclust]
8038 0000ED8E 7516 <1>      jne     short mget_w_15
8039 <1>
8040 0000ED90 8A0D[8C4E0100] <1>      mov     cl, [cdev]
8041 0000ED96 3A0D[75450100] <1>      cmp     cl, [writei.drv]
8042 0000ED9C 7508 <1>      jne     short mget_w_15
8043 <1>      ; [writei.l_clust] & [writei.l_index] are valid,
8044 <1>      ; we don't need to get last cluster & last cluster index
8045 0000ED9E 8B0D[94450100] <1>      mov     ecx, [writei.l_index]
8046 0000EDA4 EB49 <1>      jmp     short mget_w_17
8047 <1> mget_w_15:
8048 0000EDA6 A3[88450100] <1>      mov     [writei.fclust], eax ; first section (FDT number)
8049 <1>      ; edx = 0
8050 0000EDAB 8915[80450100] <1>      mov     [writei.cluster], edx ; 0 ; current section
8051 0000EDB1 8915[8C450100] <1>      mov     [writei.fs_index], edx ; 0 ; curren section index
8052 <1>
8053 <1>      ; eax = FDT number (section 0 header address)
8054 0000EDB7 E8FAD4FFFF <1>      call    get_last_section
8055 0000EDBC 0F8250FFFFFFF <1>      jc      mget_w_err ; eax = error code
8056 <1>
8057 0000EDC2 8915[8C450100] <1>      mov     [writei.fs_index], edx ; sector index in last section
8058 <1>
8059 0000EDC8 A3[90450100] <1>      mov     [writei.lclust], eax ; last section address

```

```

8060                                     <1>
8061 0000EDCD 8B0D[B4430100]             <1>      mov     ecx, [glc_index] ; last section index
8062 0000EDD3 890D[94450100]             <1>      mov     [writei.l_index], ecx
8063                                     <1>
8064 0000EDD9 A0[98450100]               <1>      mov     al, [writei.ofn]
8065 0000EDEE FEC0                       <1>      inc     al
8066 0000EDE0 A2[D3450100]               <1>      mov     [setfmod], al ; update lm date&time sign
8067                                     <1>
8068                                     <1> mget_w_16:
8069                                     <1>      ; edx = (existing) last section (sector) index
8070 0000EDE5 8B0D[84450100]             <1>      mov     ecx, [writei.c_index] ; final section (sector) index
8071 0000EDEB 29D1                       <1>      sub     ecx, edx
8072 0000EDED 7633                       <1>      jna     short mget_w_19
8073                                     <1>      ; ecx = sector count
8074                                     <1> mget_w_17:
8075 0000EDEF A1[90450100]               <1>      mov     eax, [writei.lclust]
8076                                     <1>      ; ESI = Logical dos drv desc. table address
8077                                     <1>      ; EAX = Last section
8078                                     <1>      ; (ECX = 0 for directory)
8079                                     <1>      ; ECX = sector count (except FDT)
8080 0000EDF4 E880CAFFFF                 <1>      call    add_new_fs_section
8081 0000EDF9 7312                       <1>      jnc     short mget_w_18
8082                                     <1>
8083                                     <1>      ; If error number = 27h (insufficient disk space)
8084                                     <1>      ; it is needed to check free consequent sectors
8085                                     <1>      ; (1 data sector at least and +1 section header sector)
8086                                     <1>
8087 0000EDFB 83F827                     <1>      cmp     eax, 27h
8088 0000EDFE 0F850EFFFFFF               <1>      jne     mget_w_err ; eax = error code
8089                                     <1>
8090                                     <1>      ; ecx = count of free consequent sectors
8091                                     <1>      ; ecx must be > 1 (1 data + 1 header sector)
8092 0000EE04 49                         <1>      dec     ecx
8093 0000EE05 0F8407FFFFFF               <1>      jz      mget_w_err
8094 0000EE0B EBE2                       <1>      jmp     short mget_w_17
8095                                     <1>
8096                                     <1> mget_w_18:
8097 0000EE0D A3[90450100]               <1>      mov     [writei.lclust], eax ; (new) last section
8098                                     <1>      ; ecx = sector count (except section header)
8099 0000EE12 8B15[94450100]             <1>      mov     edx, [writei.l_index]
8100 0000EE18 01CA                       <1>      add     edx, ecx ; add sector count to index
8101 0000EE1A 8915[94450100]             <1>      mov     [writei.l_index], edx
8102 0000EE20 EBC3                       <1>      jmp     short mget_w_16
8103                                     <1>
8104                                     <1> mget_w_19:
8105 0000EE22 89E9                       <1>      mov     ecx, ebp
8106 0000EE24 030D[D04E0100]             <1>      add     ecx, [u.count]
8107 0000EE2A 890D[9D4F0100]             <1>      mov     [i.size], ecx ; save new file size
8108                                     <1>      ;sub     edx, edx ; 0
8109                                     <1>
8110 0000EE30 A0[8C4E0100]               <1>      mov     al, [cdev]
8111 0000EE35 A2[75450100]               <1>      mov     [writei.drv], al ; physical drive number
8112                                     <1>      ; edx = 0
8113 0000EE3A 89E8                       <1>      mov     eax, ebp ; file offset
8114 0000EE3C 89C2                       <1>      mov     edx, eax
8115                                     <1>      ; 1 cluster = 512 bytes (for Singlix FS)
8116 0000EE3E C1E809                     <1>      shr     eax, 9 ; / 512
8117 0000EE41 81E2FF010000               <1>      and     edx, 1FFh
8118                                     <1>      ; edx = byte offset in cluster/sector (<= 511)
8119                                     <1>      ; eax = section (sector/cluster) index
8120 0000EE47 A3[84450100]               <1>      mov     [writei.c_index], eax
8121 0000EE4C 668915[7E450100]           <1>      mov     [writei.offset], dx
8122                                     <1>      ;mov     byte [writei.s_index], 0 ; sector index in cluster
8123 0000EE53 E912FEFFFFFF               <1>      jmp     mget_w_3
8124                                     <1>
8125                                     <1> update_file_lmdt: ; & update file size
8126                                     <1>      ; 26/10/2016
8127                                     <1>      ; 24/10/2016
8128                                     <1>      ; 23/10/2016
8129                                     <1>      ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)
8130                                     <1>      ;
8131                                     <1>      ; Update last modification date&time of file
8132                                     <1>      ; (call from syswrite -> writei)
8133                                     <1>      ; ((also updates file size)) // 26/10/2016
8134                                     <1>      ;
8135                                     <1>      ; INPUT:
8136                                     <1>      ;     byte [setfmod] = open file number
8137                                     <1>      ; OUTPUT:
8138                                     <1>      ;     cf = 0 -> success !
8139                                     <1>      ;     cf = 1 -> lmdt update has been failed!
8140                                     <1>      ;
8141                                     <1>      ; Modified registers: eax, ebx, ecx, edx, esi, edi
8142                                     <1>      ;
8143                                     <1>
8144                                     <1>      ;cmp     byte [setfmod], 0
8145                                     <1>      ;jna     short uflmdt_2 ; nothing to do
8146                                     <1>
8147 0000EE58 31C0                       <1>      xor     eax, eax
8148                                     <1>
8149 0000EE5A 0FB61D[D3450100]             <1>      movzx    ebx, byte [setfmod]
8150 0000EE61 FECB                       <1>      dec     bl ; open file index number (0 based)
8151                                     <1>
8152 0000EE63 8AA3[40490100]             <1>      mov     ah, [ebx+OF_DRIVE]
8153 0000EE69 BE00010900                 <1>      mov     esi, Logical_DOSDisks
8154 0000EE6E 01C6                       <1>      add     esi, eax
8155 0000EE70 C0E302                     <1>      shl     bl, 2 ; *4
8156 0000EE73 8B8B[18490100]             <1>      mov     ecx, [ebx+OF_FCLUSTER] ; first cluster
8157 0000EE79 8B93[E0490100]             <1>      mov     edx, [ebx+OF_DIRCLUSTER] ; dir cluster
8158                                     <1>
8159 0000EE7F D0EB                       <1>      shr     bl, 1 ; /2
8160 0000EE81 0FB7BB[804A0100]           <1>      movzx    edi, word [ebx+OF_DIRENTRY]
8161                                     <1>

```



```

8162 0000EE88 803D[10410100]01 <1>      cmp     byte [DirBuff_ValidData], 1
8163 0000EE8F 726E <1>      jnb     short uflmdt_4
8164 <1>
8165 0000EE91 A0[0E410100] <1>      mov     al, [DirBuff_DRV]
8166 0000EE96 2C41 <1>      sub     al, 'A'
8167 0000EE98 38E0 <1>      cmp     al, ah
8168 0000EE9A 7563 <1>      jne     short uflmdt_4 ; different drive
8169 0000EE9C 8A4603 <1>      mov     al, [esi+LD_FATType]
8170 0000EE9F 3A05[0F410100] <1>      cmp     al, [DirBuff_FATType]
8171 0000EEA5 755B <1>      jne     short uflmdt_5 ; different FS type
8172 0000EEA7 3B15[15410100] <1>      cmp     edx, [DirBuff_Cluster]
8173 0000EEAD 7553 <1>      jne     short uflmdt_5 ; different cluster
8174 <1>
8175 <1> uflmdt_1:
8176 <1>      ; Directory buffer is ready here!
8177 <1>      ; OF_FCLUSTER must be compared/verified
8178 0000EEAF BE00000800 <1>      mov     esi, Directory_Buffer
8179 0000EEB4 66C1E705 <1>      shl     di, 5 ; dir entry index * 32
8180 0000EEB8 01FE <1>      add     esi, edi ; offset
8181 <1>
8182 0000EEBA F6460B18 <1>      test    byte [esi+DirEntry_Attr], 18h ; Vol & Dir
8183 0000EEBE 750F <1>      jnz     short uflmdt_2 ; not a valid file !
8184 0000EEC0 668B4614 <1>      mov     ax, [esi+DirEntry_FstClusHI]
8185 0000EEC4 C1E010 <1>      shl     eax, 16
8186 0000EEC7 668B461A <1>      mov     ax, [esi+DirEntry_FstClusLO]
8187 0000EECB 39C8 <1>      cmp     eax, ecx ; same first cluster ?
8188 0000EECD 7407 <1>      je      short uflmdt_3 ; yes, it is OK !!!
8189 <1>
8190 <1> uflmdt_2:
8191 <1>      ; save directory buffer if has modified/changed sign
8192 <1>      ; (It is good to save dir buff even if the searched
8193 <1>      ; directory entry is not found !?)
8194 0000EECF E8F6B6FFFF <1>      call    save_directory_buffer
8195 0000EED4 F9 <1>      stc     ; update failed
8196 0000EED5 C3 <1>      retn
8197 <1>
8198 <1> uflmdt_3:
8199 <1>      ; Update directory entry
8200 <1>      ; 26/10/2016
8201 0000EED6 D0E3 <1>      shl     bl, 1 ; *2
8202 0000EED8 8B83[90490100] <1>      mov     eax, [ebx+OF_SIZE] ; file size
8203 0000EEDE 89461C <1>      mov     [esi+DirEntry_FileSize], eax
8204 <1>
8205 0000EEE1 E846B6FFFF <1>      call    convert_current_date_time
8206 <1>      ; OUTPUT -> DX = Date in dos dir entry format
8207 <1>      ; AX = Time in dos dir entry format
8208 0000EEE6 66894616 <1>      mov     [esi+DirEntry_WrtTime], ax
8209 0000EEEA 66895618 <1>      mov     [esi+DirEntry_WrtDate], dx
8210 0000EEEE 66895612 <1>      mov     [esi+DirEntry_LastAccDate], dx
8211 0000EEF2 C605[10410100]02 <1>      mov     byte [DirBuff_ValidData], 2
8212 0000EEF9 E8CCB6FFFF <1>      call    save_directory_buffer
8213 0000EEFE C3 <1>      retn
8214 <1>
8215 <1> uflmdt_4:
8216 <1>      ; Directory buffer sector read&write
8217 <1>      ; 23/10/2016
8218 <1>
8219 0000EEFF 8A4603 <1>      mov     al, [esi+LD_FATType]
8220 <1> uflmdt_5:
8221 0000EF02 BB[E4540100] <1>      mov     ebx, rw_buffer ; Common r/w sector buffer addr
8222 <1>
8223 0000EF07 20C0 <1>      and     al, al ; 0 = Singlix FS
8224 0000EF09 0F8492000000 <1>      jz      uflmdt_11
8225 <1>
8226 0000EF0F 21D2 <1>      and     edx, edx
8227 0000EF11 7521 <1>      jnz     short uflmdt_9
8228 <1>
8229 0000EF13 3C02 <1>      cmp     al, 2 ; 3 = FAT32
8230 0000EF15 771A <1>      ja      short uflmdt_8
8231 <1>
8232 0000EF17 89F8 <1>      mov     eax, edi ; directory entry index number
8233 0000EF19 66C1E804 <1>      shr     ax, 4 ; 16 entries per sector
8234 0000EF1D 034664 <1>      add     eax, [esi+LD_ROOTBegin]
8235 <1>      ; eax = root directory sector
8236 <1> uflmdt_6:
8237 0000EF20 50 <1>      push    eax ; * ; disk sector address
8238 0000EF21 51 <1>      push    ecx ; first cluster
8239 0000EF22 B901000000 <1>      mov     ecx, 1
8240 <1>      ; ecx = sector count
8241 0000EF27 E8AB020000 <1>      call    disk_read
8242 0000EF2C 59 <1>      pop     ecx
8243 0000EF2D 731A <1>      jnc     short uflmdt_10
8244 0000EF2F 58 <1>      pop     eax ; *
8245 <1> uflmdt_7:
8246 0000EF30 C3 <1>      retn
8247 <1>
8248 <1> uflmdt_8:
8249 0000EF31 8B5632 <1>      mov     edx, [esi+LD_BPB+FAT32_RootFClust]
8250 <1> uflmdt_9:
8251 0000EF34 83FA02 <1>      cmp     edx, 2
8252 0000EF37 72F7 <1>      jnb     short uflmdt_7 ; invalid, nothing to do
8253 <1>
8254 0000EF39 83EA02 <1>      sub     edx, 2
8255 0000EF3C 89D0 <1>      mov     eax, edx
8256 0000EF3E 0FB65613 <1>      movzx   edx, byte [esi+LD_BPB+SecPerClust]
8257 0000EF42 F7E2 <1>      mul     edx
8258 0000EF44 034668 <1>      add     eax, [esi+LD_DATABegin]
8259 <1>      ; eax = sub directory (data) sector
8260 0000EF47 EBD7 <1>      jmp     short uflmdt_6
8261 <1>
8262 <1> uflmdt_10:
8263 <1>      ; Directory sector buffer is ready here!

```

```

8264      <1>      ; OF_FCLUSTER must be compared/verified
8265      <1>      ; edi = dir entry index number (<= 2047)
8266 0000EF49 6683E70F      <1>      and    di, 0Fh ; 16 entries per sector
8267 0000EF4D 66C1E705      <1>      shl    di, 5 ; dir entry index * 32
8268 0000EF51 81C7[E4540100] <1>      add    edi, rw_buffer
8269      <1>      ;
8270 0000EF57 F6470B18      <1>      test   byte [edi+DirEntry_Attr], 18h ; Vol & Dir
8271 0000EF5B 0F856EFFFFFF      <1>      jnz    uflmdt_2 ; not a valid file !
8272 0000EF61 668B5714      <1>      mov     dx, [edi+DirEntry_FstClusHI]
8273 0000EF65 C1E210      <1>      shl     edx, 16
8274 0000EF68 668B571A      <1>      mov     dx, [edi+DirEntry_FstClusLO]
8275 0000EF6C 39CA      <1>      cmp     edx, ecx ; same first cluster ?
8276 0000EF6E 0F855BFFFFFF      <1>      jne     uflmdt_2 ; no !?
8277      <1>
8278      <1>      ; Update directory entry
8279 0000EF74 E8B3B5FFFF      <1>      call   convert_current_date_time
8280      <1>      ; OUTPUT -> DX = Date in dos dir entry format
8281      <1>      ;          AX = Time in dos dir entry format
8282 0000EF79 66894716      <1>      mov     [edi+DirEntry_WrtTime], ax
8283 0000EF7D 66895718      <1>      mov     [edi+DirEntry_WrtDate], dx
8284 0000EF81 66895712      <1>      mov     [edi+DirEntry_LastAccDate], dx
8285      <1>
8286 0000EF85 58      <1>      pop     eax ; *
8287      <1>
8288 0000EF86 BB[E4540100]      <1>      mov     ebx, rw_buffer ; Common r/w sector buffer addr
8289 0000EF8B B901000000      <1>      mov     ecx, 1
8290      <1>      ; esi = logical dos description table address
8291      <1>      ; eax = disk sector number/address (LBA)
8292      <1>      ; ecx = sector count
8293      <1>      ; ebx = buffer address
8294 0000EF90 E833020000      <1>      call   disk_write
8295 0000EF95 0F8234FFFFFF      <1>      jc      uflmdt_2
8296      <1>
8297      <1>      ; save directory buffer if has modified/changed sign
8298 0000EF9B E82AB6FFFF      <1>      call   save_directory_buffer
8299 0000EFA0 C3      <1>      retn
8300      <1>
8301      <1> uflmdt_11:
8302      <1>      ; 24/10/2016
8303      <1>      ; Update last modification date & time of a file
8304      <1>      ; on a disk with Singlix File System.
8305      <1>      ;
8306      <1>      ; (Method: Read the FDT -File Description Table-
8307      <1>      ; sector of the file and update the lmdt data fields,
8308      <1>      ; then write FDT sector to the disk.
8309      <1>      ; /// It is easy but there is compatibility buffer
8310      <1>      ; method also for changing directory entry data and
8311      <1>      ; also there are some programming issues for Singlix
8312      <1>      ; file system (TRFS), which are not completed yet!)
8313      <1>      ;
8314      <1>      ; Not ready yet ! (24/10/2016)
8315      <1>      ; /// Temporary code for error return ! ///
8316 0000EFA1 31C0      <1>      xor     eax, eax
8317 0000EFA3 F9      <1>      stc
8318 0000EFA4 C3      <1>      retn
8319      <1>
8320      <1> sysalloc:
8321      <1>      ; 04/03/2017
8322      <1>      ; 20/02/2017
8323      <1>      ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)
8324      <1>      ; (TRDOS 386 feature only!)
8325      <1>      ;
8326      <1>      ; Allocate Contiguous Memory Block/Pages (for user)
8327      <1>      ; (System call for DMA Buffer allocation etc.)
8328      <1>      ;
8329      <1>      ; INPUT ->
8330      <1>      ;     EBX = Virtual address (for user)
8331      <1>      ;     (Physical memory block/aperture
8332      <1>      ;     will be mapped to this virtual address)
8333      <1>      ;     ECX = Byte Count
8334      <1>      ;     (will be rounded up to page border)
8335      <1>      ;     If ECX = 0
8336      <1>      ;     System call will return with an error (cf=1)
8337      <1>      ;     but ECX will contain maximum size of
8338      <1>      ;     available memory aperture and physical
8339      <1>      ;     (beginning) address of that aperture
8340      <1>      ;     (which have maximum size) will be in EAX.
8341      <1>      ;     EDX = Upper limit of the requested physical memory
8342      <1>      ;     block/pages.
8343      <1>      ;     (The last byte address of the memory aperture
8344      <1>      ;     must not be equal to or above this limit.)
8345      <1>      ;     If EDX = 0
8346      <1>      ;     there is NOLIMIT !
8347      <1>      ;     If EDX = 0FFFFFFFh (-1)
8348      <1>      ;     ESI = Lower Limit !
8349      <1>      ;     (Beginning of the block must not be 'less'
8350      <1>      ;     than this.) (Must be equal to or above...)
8351      <1>      ;     EDI = Upper Limit !
8352      <1>      ;     (End of the block must be !less! than this)
8353      <1>      ;     (The last byte addr of the memory aperture
8354      <1>      ;     must not be equal to or above this limit.)
8355      <1>      ;
8356      <1>      ; OUTPUT ->
8357      <1>      ;     If CF = 0
8358      <1>      ;     EAX = Physical address of the allocated memory block
8359      <1>      ;     ECX = Allocated bytes (as rounded up to page borders)
8360      <1>      ;     EBX = Virtual address (as rounded up)
8361      <1>      ;     IF CF = 1
8362      <1>      ;     Requested (size of) Memory block could not be
8363      <1>      ;     allocated to the user!
8364      <1>      ;     IF CF = 1 & EAX = 0 (Insufficient memory error!)
8365      <1>      ;     ECX = Total number of free bytes

```

```

8366      <1>      ;      (not size of available contiguous bytes!)
8367      <1>      ;      If CF = 1 & EAX > 0
8368      <1>      ;      there is not a memory aperture with requested size
8369      <1>      ;      but total free mem is not less than requested size.
8370      <1>      ;      EAX = Physical addr of available memory aperture
8371      <1>      ;      with max size
8372      <1>      ;      (but it doesn't fit to the conditions!)
8373      <1>      ;      ECX = Size of available memory aperture in bytes.
8374      <1>      ;      If CF = 1 -> EAX = 0FFFFFFFh
8375      <1>      ;      Conditions/Parameters are wrong !
8376      <1>      ;      ECX is same with input value.
8377      <1>      ;
8378      <1>      ; Note:      Previously allocated pages will be deallocated if
8379      <1>      ;      new allocation conditions are met.
8380      <1>      ;
8381      <1>      ; Note: u.break control may be included in future versions
8382      <1>      ;
8383      <1>
8384      0000EFA5 31C0      <1>      xor     eax, eax ; 0
8385      0000EFA7 09D2      <1>      or      edx, edx
8386      0000EFA9 7418      <1>      jz      short sysalloc_2 ; NOLIMIT
8387      0000EFAB 48        <1>      dec     eax ; 0FFFFFFFh (-1)
8388      0000EFAC 39D1      <1>      cmp     ecx, edx
8389      0000EFAE 7743      <1>      ja      short sysalloc_wrong
8390      0000EFB0 39C2      <1>      cmp     edx, eax ; 0FFFFFFFh (-1)
8391      0000EFB2 750E      <1>      jne     short sysalloc_1
8392      0000EFB4 89FA      <1>      mov     edx, edi ; upper limit
8393      0000EFB6 29F7      <1>      sub     edi, esi
8394      0000EFB8 7639      <1>      jna     short sysalloc_wrong
8395      0000EFBA 39CF      <1>      cmp     edi, ecx ; byte count
8396      0000EFBC 7235      <1>      jb      short sysalloc_wrong
8397      0000EFBE 89F0      <1>      mov     eax, esi ; lower limit, beginning address
8398      0000EFC0 EB01      <1>      jmp     short sysalloc_2
8399      <1> sysalloc_1:
8400      0000EFC2 40        <1>      inc     eax ; 0
8401      <1> sysalloc_2:
8402      <1>      ; EAX = Beginning address (physical)
8403      <1>      ; EAX = 0 -> Allocate mem block from the 1st proper aperture
8404      <1>      ; ECX = Number of bytes to be allocated
8405      0000EFC3 E85B64FFFF      <1>      call    allocate_memory_block
8406      0000EFC8 7220      <1>      jc      short sysalloc_err
8407      <1>      ; EAX = Beginning (physical) addr of the allocated mem block
8408      <1>      ; ECX = Num of allocated bytes (rounded up to page borders)
8409      0000EFCA 50        <1>      push    eax ; * ; 04/03/2017
8410      0000EFCB 09D2      <1>      or      edx, edx ; is there a limit ?
8411      0000EFCF 7515      <1>      jnz     short sysalloc_4 ; yes
8412      <1> sysalloc_3:
8413      <1>      ; Here, requested contiguous memory pages have been allocated
8414      <1>      ; on Memory Allocation Table but user's page directory
8415      <1>      ; and page tables have not been updated yet!
8416      0000EFCF 51        <1>      push    ecx ; **
8417      <1>      ; ebx = virtual address (will be rounded up to page border)
8418      <1>      ; ecx = number of bytes to be allocated
8419      <1>      ; will be adjusted to ebx+ecx round down - ebx round up
8420      0000EFD0 E88F67FFFF      <1>      call    deallocate_user_pages
8421      0000EFD5 7326      <1>      jnc     short sysalloc_5
8422      0000EFD7 59        <1>      pop     ecx ; **
8423      0000EFD8 58        <1>      pop     eax ; *
8424      <1>      ; error
8425      <1>      ; restore Memory Allocation Table Content
8426      0000EFD9 E84666FFFF      <1>      call    deallocate_memory_block
8427      0000EFDE 31C0      <1>      xor     eax, eax ; 0
8428      0000EFE0 6648      <1>      dec     ax ; 0FFFFFFFh
8429      0000EFE2 EB0F      <1>      jmp     short sysalloc_wrong
8430      <1> sysalloc_4:
8431      <1>      ; check upper limit (edx) overlap
8432      <1>      ; ecx = byte count
8433      <1>      ; eax = beginning address (physical)
8434      0000EFE4 01C8      <1>      add     eax, ecx
8435      0000EFE6 39D0      <1>      cmp     eax, edx
8436      0000EFE8 72E5      <1>      jb      short sysalloc_3
8437      <1> sysalloc_err:
8438      0000EFEA 8B2D[A84E0100] <1>      mov     ebp, [u.usp] ; ebp points to user's registers
8439      0000EFF0 894D18      <1>      mov     [ebp+24], ecx ; return to user with ecx value
8440      <1> sysalloc_wrong:
8441      <1>      ; eax = 0FFFFFFFh
8442      0000EFF3 A3[AC4E0100] <1>      mov     [u.r0], eax
8443      0000EFF8 E9F2D3FFFF      <1>      jmp     error
8444      <1> sysalloc_5:
8445      0000EFFD 8B2D[A84E0100] <1>      mov     ebp, [u.usp] ; ebp points to user's registers
8446      0000F003 894518      <1>      mov     [ebp+24], eax ; return to user with ecx value
8447      0000F006 895D10      <1>      mov     [ebp+16], ebx ; new value of ebx
8448      0000F009 89C1      <1>      mov     ecx, eax ; byte count (from 'deallocate_user_pages')
8449      0000F00B 5A        <1>      pop     edx ; ** ; discard (another) byte count
8450      0000F00C 58        <1>      pop     eax ; *
8451      0000F00D A3[AC4E0100] <1>      mov     [u.r0], eax ; physical address
8452      <1>      ;
8453      <1>      ; Write newly allocated contiguous (physical) pages
8454      <1>      ; on page dir and page tables of current user/process
8455      <1>      ; as PRESENT, USER, WRITABLE
8456      <1>      ; (then clear allocated pages)
8457      0000F012 E84368FFFF      <1>      call    allocate_user_pages
8458      0000F017 0F83F2D3FFFF      <1>      jnc     sysret ; OK! return to process with success...
8459      <1>      ;
8460      <1>      ; unexpected error ! insufficient memory !? conflict !?
8461      <1>      ; (!?!there is not a free page for a new page table?!!)
8462      <1>      ; We need to terminate process with error message !!!
8463      <1>      ;
8464      0000F01D 8B2D[A84E0100] <1>      mov     ebp, [u.usp] ; ebp points to user's registers
8465      0000F023 8B4D18      <1>      mov     ecx, [ebp+24] ; byte count
8466      0000F026 A1[AC4E0100] <1>      mov     eax, [u.r0] ; physical address
8467      <1>      ;

```

```

8468      <1>      ; restore Memory Allocation Table Content
8469 0000F02B E8F465FFFF      <1>      call  deallocate_memory_block
8470      <1>      ;
8471 0000F030 803D[225E0000]03 <1>      cmp    byte [CRT_MODE], 3 ; 80x25 text mode?
8472 0000F037 7407      <1>      je     short sysalloc_6 ; yes
8473      <1>      ; Current mode is VGA (or CGA graphics) mode,
8474      <1>      ; We need to return to text mode for displaying
8475      <1>      ; error message just before 'sysexit'.
8476 0000F039 B003      <1>      mov    al, 3
8477 0000F03B E82525FFFF      <1>      call  _set_mode
8478      <1> sysalloc_6:
8479 0000F040 BE[09FE0000]      <1>      mov    esi, beep_Insufficient_Memory ; error message
8480 0000F045 E87372FFFF      <1>      call  print_msg ; print/display the message
8481 0000F04A B801000000      <1>      mov    eax, 1 ; ax=1 is needed for 'sysexit' procedure
8482 0000F04F E942D5FFFF      <1>      jmp    sysexit ; and terminate the process !
8483      <1>
8484      <1> sysdalloc:
8485      <1>      ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)
8486      <1>      ; (TRDOS 386 feature only!)
8487      <1>      ;
8488      <1>      ; Deallocate Memory Block/Pages (for user)
8489      <1>      ; (Complementary call for sysalloc.)
8490      <1>      ;
8491      <1>      ; INPUT ->
8492      <1>      ;     EBX = Virtual address (for user)
8493      <1>      ;           (will be rounded up to page border)
8494      <1>      ;     ECX = Byte Count
8495      <1>      ;           (will be adjusted to page borders)
8496      <1>      ;     If ICX = 0
8497      <1>      ;           nothing to do
8498      <1>      ;     If EBX + ECX > User's ESP
8499      <1>      ;           nothing to do
8500      <1>      ;
8501      <1>      ; Note: u.break control may be included in future versions
8502      <1>      ;
8503      <1>      ; OUTPUT ->
8504      <1>      ;     If CF = 0
8505      <1>      ;           EAX = Deallocated memory bytes
8506      <1>      ;           EBX = Virtual address (as rounded up)
8507      <1>      ;     IF CF = 1
8508      <1>      ;           EAX = 0
8509      <1>      ;
8510      <1>      ; Note:      Main purpose of this call is to deallocate/release
8511      <1>      ;           previously allocated (physically) contiguous memory
8512      <1>      ;           pages but beginning (virtual) address may not be
8513      <1>      ;           followed by physically contiguous pages. So, this
8514      <1>      ;           system call will deallocate user's virtually
8515      <1>      ;           contiguous memory pages. Also, there is not any
8516      <1>      ;           objections to use this system call without sysalloc
8517      <1>      ;           system call; only possible objection is to lost data
8518      <1>      ;           within user's memory space, if the beginning address
8519      <1>      ;           and size is not proper.
8520      <1>      ;
8521      <1>      ; Note: Empty page tables will not be deallocated!!!
8522      <1>      ;           (they will be deallocated at process termination)
8523      <1>      ;
8524      <1>      ; Note: When the program terminates itself or when it is
8525      <1>      ;           terminated by operating system kernel, all allocated
8526      <1>      ;           memory pages will be deallocated during termination
8527      <1>      ;           stage. So, 'sysdalloc' is not necessary except
8528      <1>      ;           forgiving memory block to other programs/processes.
8529      <1>      ;
8530 0000F054 8B15[A44E0100]      <1>      mov    edx, [u.sp]
8531 0000F05A 8B420C      <1>      mov    eax, [edx+12] ; user's stack pointer
8532 0000F05D 29C8      <1>      sub    eax, ecx ; esp - byte count
8533 0000F05F 24FC      <1>      and    al, 0FCh ; dword alignment
8534 0000F061 39D8      <1>      cmp    eax, ebx
8535 0000F063 7220      <1>      jnb    short sysdalloc_err ; deallocation overlaps with stack
8536      <1>
8537 0000F065 31C0      <1>      xor    eax, eax
8538 0000F067 21C9      <1>      and    ecx, ecx
8539 0000F069 7407      <1>      jz     short sysdalloc_2
8540      <1>
8541 0000F06B E8F466FFFF      <1>      call  deallocate_user_pages
8542 0000F070 7213      <1>      jc     short sysdalloc_err
8543      <1>
8544      <1> sysdalloc_2:
8545 0000F072 A3[AC4E0100]      <1>      mov    [u.r0], eax
8546 0000F077 8B2D[A84E0100]      <1>      mov    ebp, [u.usp]
8547 0000F07D 895D10      <1>      mov    [ebp+16], ebx ; new value of ebx
8548 0000F080 E98AD3FFFF      <1>      jmp    sysret
8549      <1>
8550      <1> sysdalloc_err:
8551 0000F085 A3[AC4E0100]      <1>      mov    [u.r0], eax ; 0
8552 0000F08A E960D3FFFF      <1>      jmp    error
8553      <1>
8554      <1> syscalbac:
8555      <1>      ; SYS CALLBACK
8556      <1>      ; 28/02/2017
8557      <1>      ; 26/02/2017
8558      <1>      ; 24/02/2017
8559      <1>      ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)
8560      <1>      ; (TRDOS 386 feature only!)
8561      <1>      ;
8562      <1>      ; Link or unlink IRQ callback service to/from user (ring 3)
8563      <1>      ;
8564      <1>      ; INPUT ->
8565      <1>      ;     BL = IRQ number (Hardware interrupt request number)
8566      <1>      ;           (0 to 15 but IRQ 0,1,2,6,8,14,15 are prohibited)
8567      <1>      ;           IRQ numbers 3,4,5,7,9,10,11,12,13 are valid
8568      <1>      ;           (numbers >15 are invalid)
8569      <1>      ;

```



```

8570      <1>      ;      BH = 0 = Unlink IRQ (in BL) from user (ring 3) service
8571      <1>      ;      1 = Link IRQ by using Signal Response Byte method
8572      <1>      ;      2 = Link IRQ by using Callback service method
8573      <1>      ;      >2 = invalid
8574      <1>      ;      (syscallback version will return to user)
8575      <1>      ;
8576      <1>      ;      CX = Signal Return/Response Byte value if < 256
8577      <1>      ;      If CX > 255
8578      <1>      ;      Kernel will put a counter value between 0 to 255
8579      <1>      ;
8580      <1>      ;      NOTE: counter value (for example: even and odd numbers
8581      <1>      ;      may be used for -audio- DMA buffer switch
8582      <1>      ;      within double buffer method, etc.)
8583      <1>      ;
8584      <1>      ;      EDX = Signal return (Response) byte address
8585      <1>      ;      - or -
8586      <1>      ;      Interrupt/Callback service/routine address
8587      <1>      ;
8588      <1>      ;      (virtual address in user's memory space)
8589      <1>      ;
8590      <1>      ; OUTPUT ->
8591      <1>      ;      CF = 0 & EAX = Successful setting
8592      <1>      ;      CF = 1 & EAX > 0 -> IRQ is prohibited or locked
8593      <1>      ;      by another process
8594      <1>      ;      AL = 1 -> Prohibited IRQ
8595      <1>      ;      AL = 2 -> IRQ already locked
8596      <1>      ;      AL = 3 -> No permission to user
8597      <1>      ;      AH = 0 -> Current 'syscallback' version
8598      <1>      ;      (structural sign of the system call)
8599      <1>      ;      (If AH=0, user will know what is possible
8600      <1>      ;      and what is impossible for current 'syscalbac'.)
8601      <1>      ;      (Next versions may have extended features and
8602      <1>      ;      less or more prohibitions.)
8603      <1>      ;
8604      <1>      ;      NOTE: Timer callbacks are set by using 'systimer'
8605      <1>      ;      system call (IRQ 0, PIT and IRQ 8, RTC)
8606      <1>      ;
8607      <1>      ;      Direct keyboard access is performed by using
8608      <1>      ;      Keyboard Interrupt (INT 32h)
8609      <1>      ;
8610      <1>      ;      It is prohibited here because:
8611      <1>      ;      1) Signal Response Byte method has not advantage
8612      <1>      ;      against INT 32h, function AH = 1. Also,
8613      <1>      ;      keyboard service interrupt will return with
8614      <1>      ;      ascii and scan codes (AL, AH) while
8615      <1>      ;      SRB method has only 1 byte space for ascii code
8616      <1>      ;      or scan code. One byte signal response is used
8617      <1>      ;      for ensuring very simple and very fast
8618      <1>      ;      virtual to physical memory address conversion
8619      <1>      ;      without any memory page crossover risk.
8620      <1>      ;      (Otherwise double page conversion or word
8621      <1>      ;      alignment would be needed.)
8622      <1>      ;      2) Badly written user code (callback code)
8623      <1>      ;      can prevent keyboard and timesharing functions
8624      <1>      ;      of the operating system via continuous and long
8625      <1>      ;      keyboard event handling by callback service.
8626      <1>      ;      (It can cause to lose immediate keystroke
8627      <1>      ;      response from hardware to user.)
8628      <1>      ;      3) If user will check any keyboard events, 'getkey'
8629      <1>      ;      (or 'getchar') must have more priority than other
8630      <1>      ;      (video etc.) events because only control ability
8631      <1>      ;      on a procedural infinite loop is a keyboard or
8632      <1>      ;      mouse event. So user can use keyboard function
8633      <1>      ;      at the end or at the beginning of a loop.
8634      <1>      ;      In this case, INT 32h is used for that purpose
8635      <1>      ;      and timer interrupt etc. callbacks can be used
8636      <1>      ;      for dynamic and synchronized data refresh/transfer
8637      <1>      ;      while cpu is in a static loop (without polling).
8638      <1>      ;      Keyboard Int callback is not more useful because
8639      <1>      ;      already a manual check (a key is pressed or not)
8640      <1>      ;      can be performed (via INT 32h, AH = 1) efficiently
8641      <1>      ;      in a loop to prevent a locked infinitive loop.
8642      <1>      ;
8643      <1>      ;      Disk IRQs (6,14,15) have been prohibited from ring 3
8644      <1>      ;      callback because, disk operations (file system services
8645      <1>      ;      etc.) are independent from user program, for fast disk r/w.
8646      <1>      ;      They are not more useful at ring 3 while they are in use
8647      <1>      ;      by standard diskio functions which are mandatory part of
8648      <1>      ;      (monolithic) OS kernel and mainprog command interpreter.
8649      <1>      ;      INT 33h diskio functions are enough for user level disk
8650      <1>      ;      r/w.
8651      <1>      ;
8652      <1>      ; TRDOS 386 - IRQ CALLBACK structures (parameters):
8653      <1>      ;
8654      <1>      ;      [u.irqlock] = 1 word, IRQ flags (0-15) that indicates
8655      <1>      ;      which IRQs are locked by (that) user.
8656      <1>      ;      Lock and unlock (by user) will change
8657      <1>      ;      these flags or 'terminate process' (sysexit)
8658      <1>      ;      will clear these flags and unlock those IRQs.
8659      <1>      ;
8660      <1>      ;      Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15
8661      <1>      ;
8662      <1>      ;      IRQ(x).owner      : 1 byte, user, [u.uno], 0 = free (unlocked)
8663      <1>      ;
8664      <1>      ;      IRQ(x).method : 1 byte for callback method & status
8665      <1>      ;      0 = Signal Response Byte method
8666      <1>      ;      1 = Callback service method
8667      <1>      ;      >1 = invalid for current 'syscallback'.
8668      <1>      ;      or(+) 80h = IRQ is in use by system (ring 0)
8669      <1>      ;      function (audio etc.) or
8670      <1>      ;      a device driver.
8671      <1>      ;      (system function will ignore the lock/owner)

```

```

8672      <1>      ;
8673      <1>      ;      IRQ(x).srb: 1 byte, Signal Return/Response byte value
8674      <1>      ;      (a fixed value by user or a counter value
8675      <1>      ;      from 0 to 255, which is increased by every
8676      <1>      ;      interrupt just before putting it into
8677      <1>      ;      the Signal Response byte address
8678      <1>      ;      (This is not used in callback serv method)
8679      <1>      ;
8680      <1>      ;      IRQ(x).addr      : 1 dword
8681      <1>      ;      Signal Response Byte address (physical)
8682      <1>      ;      -or-
8683      <1>      ;      Callback service address (virtual)
8684      <1>      ;
8685      <1>      ;      IRQ(x).dev: 1 byte
8686      <1>      ;      0 = Default device or kernel function
8687      <1>      ;      -or-
8688      <1>      ;      1-255 = Assigned device driver number
8689      <1>      ;
8690      <1>      ;      (x) = 3,4,5,7,9,10,11,12,13
8691      <1>      ;
8692      <1>      ;
8693      <1>      ;      NOTE: If user's process/program calls the kernel (INT 40h)
8694      <1>      ;      while it is already running in a (ring 3) callback
8695      <1>      ;      service, kernel will force (convert) system call to
8696      <1>      ;      'sysrele' (sys release). So, this feature provides
8697      <1>      ;      easy and simple usage of callback services without
8698      <1>      ;      falling into deepless <please 'callback me' then
8699      <1>      ;      let me 'callback you'> cycles! (User must return
8700      <1>      ;      from callback service by using 'sysrele' system
8701      <1>      ;      call, without a significant delay. Otherwise user
8702      <1>      ;      process/program may be late to catch the next event
8703      <1>      ;      within same callback purpose.
8704      <1>      ;
8705      <1>      ;
8706      0000F08F 80FB0F      <1>      cmp     bl, 15
8707      0000F092 773E      <1>      ja     short syscb_3
8708      <1>      ;
8709      0000F094 80FF02      <1>      cmp     bh, 2
8710      0000F097 7739      <1>      ja     short syscb_3 ; invalid parameter
8711      <1>      ;
8712      0000F099 0FB6FB      <1>      movzx   edi, bl ; save IRQ number
8713      <1>      ;
8714      <1>      ;      IRQ 0,1,2,6,8,14,15 are prohibited
8715      <1>      ;      ;IRQenum: ; 'trdosk9.s'
8716      <1>      ;      ;      db  0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0
8717      <1>      ;
8718      0000F09C 0FB6B7[DC030100] <1>      movzx   esi, byte [edi+IRQenum] ; IRQ availability
8719      <1>      ;      ; enumeration/index
8720      0000F0A3 08FF      <1>      or      bh, bh
8721      0000F0A5 7439      <1>      jz     short syscb_4 ; unlink the IRQ (in BL)
8722      <1>      ;
8723      0000F0A7 FECF      <1>      dec     bh
8724      <1>      ;      ; bh = method (0 = signal response byte, 1 = callback)
8725      0000F0A9 883D[F44A0100] <1>      mov     [rcallback], bh
8726      <1>      ;
8727      <1>      ;      ;dec     esi
8728      0000F0AF 664E      <1>      dec     si
8729      0000F0B1 780B      <1>      js     short syscb_1 ; 0 -> 0FFFFh
8730      <1>      ;
8731      <1>      ;      ; ESI = IRQ callback parameters index number (0 to 8)
8732      <1>      ;
8733      0000F0B3 80BE[F84A0100]00 <1>      cmp     byte [esi+IRQ.owner], 0 ; locked ?
8734      0000F0BA 765E      <1>      jna     short syscb_5 ; no... OK...
8735      <1>      ;
8736      <1>      syscb_0:
8737      <1>      ;      ; permission denied (prohibited IRQ)
8738      <1>      ;      ;mov     eax, 1
8739      0000F0BC B001      <1>      mov     al, 1 ; Phohibed IRQ
8740      <1>      syscb_1:
8741      0000F0BE C705[104F0100]0B00- <1>      mov     dword [u.error], ERR_PERM_DENIED
8741      0000F0C6 0000      <1>      ;
8742      <1>      syscb_2:
8743      0000F0C8 A3[AC4E0100]      <1>      mov     [u.r0], eax
8744      0000F0CD E91DD3FFFF      <1>      jmp     error
8745      <1>      syscb_3:
8746      0000F0D2 C705[104F0100]1700- <1>      mov     dword [u.error], ERR_INV_PARAMETER
8746      0000F0DA 0000      <1>      ;
8747      0000F0DC 31C0      <1>      xor     eax, eax ; 0 = invalid parameters & syscalbac version = 1
8748      <1>      ;      ; (0 = sys callback version 1 or revision 0)
8749      0000F0DE EBE8      <1>      jmp     short syscb_2
8750      <1>      ;
8751      <1>      syscb_4: ; unlink the requested IRQ (if it belongs to current user)
8752      0000F0E0 8A86[F84A0100] <1>      mov     al, [esi+IRQ.owner]
8753      0000F0E6 3A05[FB4E0100] <1>      cmp     al, [u.uno]
8754      <1>      ;jne     short syscb_9 ; nothing to do !!!
8755      0000F0EC 75E4      <1>      jne     short syscb_3
8756      <1>      ;
8757      0000F0EE FE0D[1E4F0100] <1>      dec     byte [u.irqc] ; decrease IRQ count (in use)
8758      <1>      ;
8759      0000F0F4 28C0      <1>      sub     al, al
8760      0000F0F6 8886[F84A0100] <1>      mov     [esi+IRQ.owner], al ; 0 ; free !!!
8761      <1>      ;and     byte [esi+IRQ.method], 80h
8762      <1>      ;mov     [esi+IRQ.srb], al ; 0
8763      <1>      ;mov     [esi+IRQ.dev], al ; 0
8764      <1>      ;mov     [esi+IRQ.addr], eax ; 0
8765      0000F0FC A3[AC4E0100] <1>      mov     [u.r0], eax
8766      <1>      ;
8767      <1>      ; 28/02/2017
8768      0000F101 803D[0A4B0100]80 <1>      cmp     byte [IRQ.method], 80h ; device drv or kernel extension ?
8769      0000F108 0F8301D3FFFF      <1>      jnb     sysret ; yes, do not reset to default IRQ handler
8770      <1>      ;
8771      0000F10E 88D8      <1>      mov     al, bl ; IRQ number

```

```

8772      <1>      ; ah = 0 ; reset to default interrupt handler (ilist)
8773 0000F110 E810060000      <1>      call  set_hardware_int_vector
8774      <1>
8775 0000F115 E9F5D2FFFF      <1>      jmp    sysret
8776      <1>
8777      <1> syscb_5:
8778 0000F11A A0[FB4E0100]      <1>      mov    al, [u.uno] ; user (process) number (1 to 16)
8779 0000F11F 8886[F84A0100]      <1>      mov    [esi+IRQ.owner], al ; lock the IRQ for user
8780 0000F125 8A86[0A4B0100]      <1>      mov    al, [esi+IRQ.method] ; >= 80h = kernel is using this IRQ
8781 0000F12B 2480      <1>      and    al, 80h ; use only bit 7 (kernel function flag)
8782 0000F12D 0A05[F44A0100]      <1>      or     al, [rcallback] ; method
8783      <1> ; 0 = signal response byte, 1 = callback
8784 0000F133 A801      <1>      test   al, 1
8785 0000F135 7509      <1>      jnz    short syscb_6 ; Callback method, CX will not be used
8786      <1>
8787 0000F137 6681F9FF00      <1>      cmp    cx, 255 ; If CX > 255, use counter method
8788 0000F13C 7604      <1>      jna    short syscb_7 ; (count can be used for buffer switch)
8789 0000F13E 0C02      <1>      or     al, 2
8790      <1> syscb_6:
8791 0000F140 31C9      <1>      xor    ecx, ecx ; 0
8792      <1> syscb_7:
8793 0000F142 8886[0A4B0100]      <1>      mov    [esi+IRQ.method], al
8794 0000F148 888E[134B0100]      <1>      mov    [esi+IRQ.srb], cl
8795 0000F14E C686[014B0100]00      <1>      mov    byte [esi+IRQ.dev], 0 ; device number is always 0
8796      <1> ; for this system call
8797 0000F155 A801      <1>      test   al, 1
8798 0000F157 7517      <1>      jnz    short syscb_8 ; callback method, use virtual address
8799      <1>
8800 0000F159 53      <1>      push   ebx ; IRQ num
8801 0000F15A 89D3      <1>      mov    ebx, edx
8802      <1> ; ebx = virtual address
8803      <1> ; [u.pgdir] = page directory's physical address
8804 0000F15C FE05[F64A0100]      <1>      inc    byte [no_page_swap] ; 1
8805      <1> ; Do not add this page to swap queue
8806      <1> ; and remove it from swap queue if it is
8807      <1> ; on the queue.
8808 0000F162 E82761FFFF      <1>      call   get_physical_addr
8809 0000F167 5B      <1>      pop    ebx
8810 0000F168 0F8264FFFFFFF      <1>      jc     syscb_3 ; invalid address !
8811      <1> ; eax = physical address of the virtual address in user's space
8812 0000F16E 89C2      <1>      mov    edx, eax
8813      <1> syscb_8:
8814 0000F170 66C1E602      <1>      shl    si, 2 ; byte (index) to dword (offset)
8815 0000F174 8996[1C4B0100]      <1>      mov    [esi+IRQ.addr], edx
8816      <1>
8817 0000F17A FE05[1E4F0100]      <1>      inc    byte [u.irqc] ; increase IRQ (in use) count
8818      <1> syscb_9:
8819 0000F180 31C0      <1>      xor    eax, eax
8820 0000F182 A3[AC4E0100]      <1>      mov    [u.r0], eax ; 0
8821      <1>
8822 0000F187 88D8      <1>      mov    al, bl ; IRQ number
8823 0000F189 FEC4      <1>      inc    ah ; set to requested IRQ handler (IRQ_u_list)
8824 0000F18B E895050000      <1>      call   set_hardware_int_vector
8825      <1>
8826 0000F190 E97AD2FFFF      <1>      jmp    sysret ; return with success (cf=0, eax=0)
8827      <1>
8828      <1>
8829      <1> sysfpstat:
8830      <1> ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)
8831      <1> ; (TRDOS 386 feature only!)
8832      <1> ;
8833      <1> ; Set or reset FPU registers save/restore option (for user)
8834      <1> ; (during software task switching, wswap-rswap)
8835      <1> ;
8836      <1> ; INPUT ->
8837      <1> ; BL = 0 -> reset
8838      <1> ; BL = 1 -> set (FPU register will be saved and restored)
8839      <1> ;
8840      <1> ; OUTPUT ->
8841      <1> ; cf = 0 -> no error, FPU is ready...
8842      <1> ; (EAX = 0)
8843      <1> ; Cf = 1 -> error, 80387 FPU is not ready !
8844      <1> ; (EAX = 0FFFFFFFh)
8845      <1>
8846 0000F195 31C0      <1>      xor    eax, eax
8847 0000F197 803D[E0450100]00      <1>      cmp    byte [fpready], 0
8848 0000F19E 7613      <1>      jna    short sysfpstat_err
8849      <1>
8850 0000F1A0 80E301      <1>      and    bl, 1 ; use BIT 0 only !
8851 0000F1A3 881D[224F0100]      <1>      mov    [u.fpsave], bl
8852 0000F1A9 A3[AC4E0100]      <1>      mov    [u.r0], eax ; 0
8853 0000F1AE E95CD2FFFF      <1>      jmp    sysret
8854      <1>
8855      <1> sysfpstat_err:
8856 0000F1B3 48      <1>      dec    eax ; 0FFFFFFFh
8857 0000F1B4 A3[AC4E0100]      <1>      mov    [u.r0], eax ; -1
8858 0000F1B9 E931D2FFFF      <1>      jmp    error
8859      <1>
8860      <1>
8861      <1> ;maknod:
8862      <1> ; 26/10/2016
8863      <1> ; temporary
8864      <1> ; retn
8865      <1>
8866      <1> ; temporary - 24/01/2016
8867      <1>
8868      <1> iget:
8869 0000F1BE C3      <1>      retn
8870      <1> isintr:
8871 0000F1BF C3      <1>      retn
8872      <1> iopen:
8873 0000F1C0 C3      <1>      retn

```

```

8874      <1> iclose:
8875      <1>         retn
8876      <1> settimod:
8877      <1>         retn
8878      <1> sndc:
8879      <1>         retn
8880      <1> access:
8881      <1>         retn
8882      <1> epoch:
8883      <1>         retn
8884      <1> sleep:
8885      <1>         retn
8886      <1> set_date_time:
8887      <1>         retn
2310      %include 'trdosk7.s' ; 24/01/2016
      <1> ; *****
      <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - DISK READ&WRITE : trdosk7.s
      <1> ; -----
      <1> ; Last Update: 25/02/2016
      <1> ; -----
      <1> ; Beginning: 24/01/2016
      <1> ; -----
      <1> ; Assembler: NASM version 2.11 (trdos386.s)
      <1> ; -----
      <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
      <1> ; DISK_IO.ASM (20/07/2011)
      <1> ; *****
      <1> ; DISK_IO.ASM (c) 2009-2011 Erdogan TAN [ 04/07/2009 ] Last Update: 20/07/2011
      <1>
      <1> disk_write:
      <1>         ; 25/02/2016
      <1>         ; 24/02/2016
      <1>         ; 23/02/2016
      <1>         cmp     byte [esi+LD_LBAYes], 0
      <1>         ja      short lba_write
      <1>
      <1> chs_write:
      <1>         ; 25/02/2016
      <1>         ; 23/02/2016
      <1>         mov     byte [disk_rw_op], 3 ; CHS write
      <1>         jmp     short chs_rw
      <1>
      <1> disk_read:
      <1>         ; 25/02/2016
      <1>         ; 24/02/2016
      <1>         ; 23/02/2016
      <1>         ; 17/02/2016
      <1>         ; 14/02/2016
      <1>         ; 31/01/2016 (TRDOS 386 = TRDOS v2.0)
      <1>         ; 17/10/2010
      <1>         ; 18/04/2010
      <1>         ;
      <1>         ; INPUT -> EAX = Logical Block Address
      <1>         ;         ESI = Logical Dos Disk Table Offset (DRV)
      <1>         ;         ECX = Sector Count
      <1>         ;         EBX = Destination Buffer
      <1>         ; OUTPUT ->
      <1>         ;         cf = 0 or cf = 1
      <1>         ; (Modified registers: EAX, EBX, ECX, EDX)
      <1>
      <1>         cmp     byte [esi+LD_LBAYes], 0
      <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
      <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:
      <1>         mov     byte [retry_count], 4
      <1>
      <1> chs_read_retry:
      <1>         ;mov     [sector_count], ecx ; 23/02/2016
      <1>
      <1>         push     eax                ; Linear sector #
      <1>         push     ecx                ; # of FAT/FILE/DIR sectors
      <1>
      <1>         movzx    ecx, word [esi+LD_BPB+SecPerTrack]
      <1>         ;movzx    ecx, byte [disk_rw_spt] ; 23/02/2016
      <1>         sub      edx, edx
      <1>         div      ecx
      <1>         ; eax = track, dx (dl ) = sector (on track)
      <1>         ;sub     cl, dl ; 24/02/2016 (spt - sec)

```



```

88      <1>      ;push ecx ; *
89      <1>      mov cx, dx          ; Sector (zero based)
90      <1>      inc cx              ; To make it 1 based
91      <1>      push cx
92      <1>      mov cx, [esi+LD_BPB+Heads]
93      <1>      sub dx, dx
94      <1>      div ecx              ; Convert track to head & cyl
95      <1>      ; eax (ax) = cylinder, dx (dl) = head (max. FFh)
96      <1>      mov dh, dl
97      <1>      pop cx              ; AX=Cyl, DH=Head, CX=Sector
98      <1>      mov dl, [esi+LD_PhyDrvNo]
99      <1>
100     <1>      mov ch, al          ; NOTE: max. 1023 cylinders !
101     <1>      ror ah, 2           ; Rotate 2 bits right
102     <1>      or cl, ah
103     <1>
104     <1>      ; 24/02/2016
105     <1>      ;pop eax ; * (spt - sec) (example: 63 - 0 = 63)
106     <1>      ;cmp eax, [sector_count]
107     <1>      ;jb short chs_write_sectors
108     <1>      ;je short chs_read_sectors
109     <1>      ; ; (# of sectors to read is more than remaining sectors on the track)
110     <1>      ;mov al, [sector_count]
111     <1>      ;chs_read_sectors: ; read or write !
112     <1>      mov al, 1 ; 25/02/2016
113     <1>      mov ah, [disk_rw_op] ; 02h = chs read, 03h = chs write
114     <1>      ;
115     <1>      call int13h          ; BIOS Service func ( ah ) = 2
116     <1>      ; Read disk sectors
117     <1>      ; AL-sec num CH-track CL-sec
118     <1>      ; DH-head DL-drive ES:BX-buffer
119     <1>      ; CF-flag AH-stat AL-sec read
120     <1>      ; If CF = 1 then (If AH > 0)
121     <1>      mov [disk_rw_err], ah
122     <1>
123     <1>      pop ecx
124     <1>      pop eax
125     <1>      jnc short chs_read_ok
126     <1>
127     <1>      cmp byte [disk_rw_err], 09h ; DMA crossed 64K segment boundary
128     <1>      je short chs_read_error_retn
129     <1>
130     <1>      dec byte [retry_count]
131     <1>      jnz short chs_read_retry
132     <1>
133     <1>      chs_read_error_retn:
134     <1>      stc
135     <1>      ;retn
136     <1>      jmp short update_drv_error_byte
137     <1>
138     <1>      ;chs_write_sectors: ; read or write
139     <1>      ; ; (# of sectors to read is less than remaining sectors on the track)
140     <1>      ;mov [sector_count], al
141     <1>      ;jmp short chs_read_sectors
142     <1>
143     <1>      chs_read_ok:
144     <1>      ; ; 23/02/2016
145     <1>      ;movzx edx, byte [sector_count] ; sector count (<= spt)
146     <1>      ;sub ecx, edx ; remaining sector count
147     <1>      ;jna short update_drv_error_byte
148     <1>      ;add eax, edx ; next disk sector
149     <1>      ;shl edx, 9 ; 512 * sector count
150     <1>      ;add ebx, edx ; next buffer byte address
151     <1>      ;jmp chs_read_next_sector
152     <1>      ; 25/02/2016
153     <1>      inc eax ; next sector
154     <1>      add ebx, 512
155     <1>      loop chs_read_next_sector
156     <1>      jmp short update_drv_error_byte
157     <1>
158     <1>      lba_write:
159     <1>      ; 23/02/2016
160     <1>      mov byte [disk_rw_op], 1Ch ; LBA write
161     <1>      jmp short lba_rw
162     <1>
163     <1>      lba_read:
164     <1>      ; 23/02/2016
165     <1>      ; 17/02/2016
166     <1>      ; 14/02/2016
167     <1>      ; 13/02/2016
168     <1>      ; 31/01/2016 (TRDOS 386 = TRDOS v2.0)
169     <1>      ; 10/07/2015 (Retro UNIX 386 v1)
170     <1>      ;
171     <1>      ; INPUT -> EAX = Logical Block Address
172     <1>      ; ESI = Logical Dos Disk Table Offset (DRV)
173     <1>      ; ECX = Sector Count
174     <1>      ; EBX = Destination Buffer
175     <1>      ; OUTPUT ->
176     <1>      ; cf = 0 or cf = 1
177     <1>      ; (Modified registers: EAX, EBX, ECX, EDX)
178     <1>
179     <1>      ; LBA read/write (with private LBA function)
180     <1>      ; ((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
181     <1>
182     <1>
183     <1>      ; 23/02/2016
184     <1>      mov byte [disk_rw_op], 1Bh ; LBA read
185     <1>
186     <1>      lba_rw:
187     <1>      ; 17/02/2016
188     <1>      push edi
189     <1>

```

```

190 0000F25A 890D[DC410100] <1>      mov     [sector_count], ecx ; total sector (read) count
191                                <1>
192 0000F260 8A5602          <1>      mov     dl, [esi+LD_PhyDrvNo]
193                                <1>      ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
194                                <1>
195                                <1> lba_read_next:
196 0000F263 81F900010000    <1>      cmp     ecx, 256
197 0000F269 7605          <1>      jna     short lba_read_rsc
198 0000F26B B900010000    <1>      mov     ecx, 256 ; 17/02/2016
199                                <1> lba_read_rsc:
200 0000F270 290D[DC410100] <1>      sub     [sector_count], ecx ; remain sectors
201                                <1>
202 0000F276 89CF          <1>      mov     edi, ecx
203 0000F278 89C1          <1>      mov     ecx, eax ; sector number/address
204                                <1>
205 0000F27A C605[DA410100]04 <1>      mov     byte [retry_count], 4
206                                <1> lba_read_retry:
207 0000F281 89F8          <1>      mov     eax, edi
208                                <1>      ;
209                                <1>      ; ecx = sector number
210                                <1>      ; al = sector count (0 - 255) /// (0 = 256)
211                                <1>      ; dl = drive number
212                                <1>      ; ebx = buffer offset
213                                <1>      ;
214                                <1>      ; Function 1Bh = LBA read, 1Ch = LBA write
215                                <1>      ; 23/02/2016
216 0000F283 8A25[D9410100] <1>      mov     ah, [disk_rw_op] ; 1Bh = LBA read, 1Ch = LBA write
217 0000F289 E8784FFFFFFF    <1>      call    int13h
218                                <1>      ; al = ? (changed)
219                                <1>      ; ah = error code
220 0000F28E 8825[DB410100] <1>      mov     [disk_rw_err], ah
221 0000F294 7334          <1>      jnc     short lba_read_ok
222 0000F296 80FC80          <1>      cmp     ah, 80h ; time out?
223 0000F299 740A          <1>      je      short lba_read_stc_retn
224 0000F29B FE0D[DA410100] <1>      dec     byte [retry_count]
225 0000F2A1 7FDE          <1>      jg     short lba_read_retry
226 0000F2A3 743A          <1>      jz     short lba_read_reset
227                                <1>      ; sf = 1
228                                <1>
229                                <1> lba_read_stc_retn:
230 0000F2A5 F9            <1>      stc
231                                <1> lba_read_retn:
232 0000F2A6 5F            <1>      pop     edi
233                                <1>
234                                <1> update_drv_error_byte:
235 0000F2A7 9C            <1>      pushf
236 0000F2A8 53            <1>      push     ebx
237 0000F2A9 6651          <1>      push     cx
238                                <1>      ;or     ecx, ecx
239                                <1>      ;jz     short udrv_errb0
240 0000F2AB 8A0D[DB410100] <1>      mov     cl, [disk_rw_err]
241                                <1> udrv_errb0:
242 0000F2B1 0FB65E02      <1>      movzx   ebx, byte [esi+LD_PhyDrvNo]
243 0000F2B5 80FB02          <1>      cmp     bl, 2
244 0000F2B8 7203          <1>      jb     short udrv_errb1
245 0000F2BA 80EB7E          <1>      sub     bl, 7Eh
246                                <1>      ;cmp    bl, 5
247                                <1>      ;ja     short udrv_errb2
248                                <1> udrv_errb1:
249 0000F2BD 81C3[A95C0000] <1>      add     ebx, drv.error ; 13/02/2016
250 0000F2C3 880B          <1>      mov     [ebx], cl ; error code
251                                <1> udrv_errb2:
252 0000F2C5 6659          <1>      pop     cx
253 0000F2C7 5B            <1>      pop     ebx
254 0000F2C8 9D            <1>      popf
255 0000F2C9 C3            <1>      retn
256                                <1>
257                                <1> lba_read_ok:
258 0000F2CA 89C8          <1>      mov     eax, ecx ; sector number
259 0000F2CC 01F8          <1>      add     eax, edi ; sector number (next)
260 0000F2CE C1E709          <1>      shl     edi, 9 ; sector count * 512
261 0000F2D1 01FB          <1>      add     ebx, edi ; next buffer offset
262                                <1>
263 0000F2D3 8B0D[DC410100] <1>      mov     ecx, [sector_count] ; remaining sectors
264 0000F2D9 09C9          <1>      or      ecx, ecx
265 0000F2DB 7586          <1>      jnz     short lba_read_next
266 0000F2DD EBC7          <1>      jmp     short lba_read_retn
267                                <1>
268                                <1> lba_read_reset:
269 0000F2DF B40D          <1>      mov     ah, 0Dh ; Alternate reset
270 0000F2E1 E8204FFFFFFF    <1>      call    int13h
271                                <1>      ; al = ? (changed)
272                                <1>      ; ah = error code
273 0000F2E6 7399          <1>      jnc     short lba_read_retry
274 0000F2E8 EBBC          <1>      jmp     short lba_read_retn
2311 %include 'trdosk8.s' ; 24/01/2016
1  <1> ; *****
2  <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - MAIN PROGRAM : trdosk8.s
3  <1> ; -----
4  <1> ; Last Update: 18/03/2017
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> ; u0.s (20/11/2015), u4.s (14/10/2015)
12 <1> ; *****
13 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
14 <1> ; TRDOS2.ASM (09/11/2011)
15 <1> ; -----
16 <1> ; DIR.ASM (c) 2004-2011 Erdogan TAN [07/01/2004] Last Update: 09/10/2011

```

```

17      <1>
18      <1> set_run_sequence:
19      <1>      ; 23/12/2016
20      <1>      ; 10/06/2016
21      <1>      ; 22/05/2016
22      <1>      ; 20/05/2016
23      <1>      ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
24      <1>      ; TRDOS 386 feature only !
25      <1>      ;
26      <1>      ; INPUT ->
27      <1>      ;     AL = process number (next process)
28      <1>      ;
29      <1>      ;     this process must be added to run sequence
30      <1>      ;
31      <1>      ;     [u.pri] = priority of present process
32      <1>      ;
33      <1>      ;     DL = priority (queue)
34      <1>      ;         0 = background (low) ; run on background
35      <1>      ;         1 = regular (normal) ; run as regular
36      <1>      ;         2 = event (high)      ; run for event
37      <1>      ;
38      <1>      ;     1) If the requested process is already running:
39      <1>      ;         a) If present priority is high ([u.pri]=2)
40      <1>      ;             and requested priority is also high,
41      <1>      ;             there is nothing to do! Because it has been
42      <1>      ;             done already (before this attempt).
43      <1>      ;         b) If present priority is high ([u.pri]=2)
44      <1>      ;             and requested priority is not high, there is
45      <1>      ;             nothing to do! Because, it's current
46      <1>      ;             run queue is unspecified, here. (It may be in
47      <1>      ;             a waiting list or in a run queue; if the new
48      <1>      ;             priority would be used to add it to relevant
49      <1>      ;             run queue, this would be wrong, unnecessary
50      <1>      ;             and destabilizing duplication!)
51      <1>      ;         c) If present priority is not high ([u.pri]<2)
52      <1>      ;             and requested priority is high (event),
53      <1>      ;             process will be added to present priority's
54      <1>      ;             run queue and then, priority will be changed
55      <1>      ;             to high ([u.pri]=2).
56      <1>      ;         d) If present priority is not high ([u.pri]<2)
57      <1>      ;             and requested priority is not high, [u.pri]
58      <1>      ;             value will be changed. There is nothing to do
59      <1>      ;             in addition. (The new priority value will be
60      <1>      ;             used by 'tswap/tswitch' procedure at 'sysret'
61      <1>      ;             or 'sysrele' stage.)
62      <1>      ;
63      <1>      ;     2) If the requested process is not running:
64      <1>      ;         a) If requested priority of the requested
65      <1>      ;             (next) process is high (event) and priority
66      <1>      ;             of present process is not high, the requested
67      <1>      ;             process will be added to ('runq_event') high
68      <1>      ;             priority run queue and then present (running)
69      <1>      ;             process will be stopped (swapped/switched out)
70      <1>      ;             immediately if it is in user mode, or it's
71      <1>      ;             [u.quant] value will be reset to 0 and (then)
72      <1>      ;             it will be stopped at 'sysret' stage.
73      <1>      ;         b) If requested priority of the requested
74      <1>      ;             (next) process is high (event) and priority
75      <1>      ;             of present process is also high, the requested
76      <1>      ;             process will be added to ('runq_event') high
77      <1>      ;             priority run queue and present (running)
78      <1>      ;             process will be allowed to run until it's
79      <1>      ;             time quantum will be elapsed ([u.quant]=0).
80      <1>      ;         c) If requested priority of the requested
81      <1>      ;             (next) process is not high ('run for event'),
82      <1>      ;             there is nothing to do. Because, it's current
83      <1>      ;             run queue is unspecified, here. (It may be in
84      <1>      ;             a waiting list or in a run queue; if the new
85      <1>      ;             priority would be used to add it to relevant
86      <1>      ;             run queue, this would be wrong, unnecessary
87      <1>      ;             and destabilizing duplication!)
88      <1>      ;
89      <1>      ; OUTPUT ->
90      <1>      ;     none
91      <1>      ;
92      <1>      ;     [u.pri] = priority of present process
93      <1>      ;
94      <1>      ;     cf = 1, if the request could not be fulfilled.
95      <1>      ;
96      <1>      ;     NOTE:
97      <1>      ;         * Processes in 'run as regular' queue can run
98      <1>      ;         if there is no process in 'run for event' queue
99      <1>      ;         ('run for event' processes have higher priority)
100     <1>      ;         * When [u.quant] time quantum of a process is
101     <1>      ;         elapsed, it's high priority ('run for event')
102     <1>      ;         status will be disabled, it can be run in sequence
103     <1>      ;         of it's actual run queue.
104     <1>      ;         * A 'run on background' process will always be
105     <1>      ;         sequenced in 'run on background' (low priority)
106     <1>      ;         queue, it can run only when other priority queues
107     <1>      ;         are empty. (idle time processes, e.g. printing)
108     <1>      ;
109     <1>      ; Modified registers: eax, ebx, edx
110     <1>      ;
111     <1>
112     <1> srunseq_0:
113 0000F2EA 3A05[FB4E0100] <1>      cmp     al, [u.uno]      ; same process ?
114 0000F2F0 750C <1>      jne     short srunseq_2 ; no
115 <1>
116 0000F2F2 8A25[F14E0100] <1>      mov     ah, [u.pri]      ; present/current priority
117 0000F2F8 80FC02 <1>      cmp     ah, 2           ; 'run for event' priority level
118 0000F2FB 7221 <1>      jnb     short srunseq_6 ; no

```

```

119 <1>
120 <1> srunseq_1:
121 <1> ; there is nothing to do!
122 0000F2FD C3 <1> retn
123 <1>
124 <1> srunseq_2:
125 <1> ;this not necessary ! 23/12/2016
126 <1> ;cmp al, nproc ; number of processes = 16
127 <1> ;jnb short srunseq_5 ; error ! invalid process number
128 <1>
129 <1> ; dl = priority
130 0000F2FE 80FA02 <1> cmp dl, 2 ; event queue
131 0000F301 72FA <1> jb short srunseq_1 ; requested process is not present
132 <1> ; process and priority of requested
133 <1> ; process is not high (event),
134 <1> ; there is nothing to do!
135 <1>
136 <1> ; requested process is not present process
137 <1> ; & priority of requested process is high
138 0000F303 3A15[F14E0100] <1> cmp dl, [u.pri] ; priority of present process
139 0000F309 7606 <1> jna short srunseq_3 ; is high, also
140 <1> ;
141 <1> ; present process will be swapped/switched out
142 0000F30B FE05[B5450100] <1> inc byte [p_change] ; 1
143 <1>
144 <1> srunseq_3:
145 <1> ; add process to 'runq_event' queue for new event
146 0000F311 BB[984E0100] <1> mov ebx, runq_event ; high priority run queue
147 <1>
148 <1> srunseq_4:
149 <1> ; al = process number
150 <1> ; ebx = run queue
151 0000F316 E8CAF3FFFF <1> call putlu
152 0000F31B C3 <1> retn
153 <1>
154 <1> srunseq_5:
155 0000F31C F5 <1> cmc
156 0000F31D C3 <1> retn
157 <1>
158 <1> srunseq_6:
159 <1> ; present priority of the process is not high
160 <1>
161 0000F31E 8815[F14E0100] <1> mov [u.pri], dl ; new priority
162 <1> ; (will be used by 'tswap')
163 <1>
164 0000F324 80FA02 <1> cmp dl, 2 ; high priority ?
165 0000F327 72F3 <1> jb short srunseq_5 ; no, there is nothing to do
166 <1> ; in addition
167 <1>
168 <1> ; process must be added to relevant run queue, here!
169 <1> ; (new priority is high/event priority and process
170 <1> ; will not be added to a run queue by 'tswap')
171 <1>
172 0000F329 BB[9A4E0100] <1> mov ebx, runq_normal ; 'run as regular' queue
173 <1>
174 0000F32E 20E4 <1> and ah, ah ; previous value of [u.pri]
175 0000F330 75E4 <1> jnz short srunseq_4
176 <1>
177 0000F332 43 <1> inc ebx
178 0000F333 43 <1> inc ebx
179 <1> ; ebx = runq_background ; 'run on backgroud' queue
180 <1>
181 0000F334 EBE0 <1> jmp short srunseq_4
182 <1> clock:
183 <1> ; 23/05/2016
184 <1> ; 22/05/2016
185 <1> ; 20/05/2016
186 <1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
187 <1> ; 14/05/2015 - 14/10/2015 (Retro UNIX 386 v1)
188 <1> ; 07/12/2013 - 10/04/2014 (Retro UNIX 8086 v1)
189 <1>
190 0000F336 803D[F04E0100]00 <1> cmp byte [u.quant], 0
191 0000F33D 772C <1> ja short clk_1
192 <1> ;
193 0000F33F 803D[FB4E0100]01 <1> cmp byte [u.uno], 1 ; /etc/init ? (for Retro UNIX 8086 & 386 v1)
194 <1> ; MainProg (Kernel's Command Interpreter)
195 <1> ; for TRDOS 386.
196 0000F346 7623 <1> jna short clk_1 ; yes, do not swap out
197 <1> ;
198 0000F348 803D[A14E0100]FF <1> cmp byte [sysflg], 0FFh ; user or system space ?
199 0000F34F 7520 <1> jne short clk_2 ; system space (sysflg <> 0FFh)
200 <1> ;
201 0000F351 66833D[F24E0100]00 <1> cmp word [u.intr], 0
202 0000F359 7616 <1> jna short clk_2
203 <1> ;
204 <1> ; 23/05/2016
205 0000F35B 803D[B6450100]00 <1> cmp byte [multi_tasking], 0
206 0000F362 760D <1> jna short clk_2
207 <1> ;
208 0000F364 FE05[B5450100] <1> inc byte [p_change] ; it is time to change running process
209 0000F36A C3 <1> retn
210 <1> clk_1:
211 0000F36B FE0D[F04E0100] <1> dec byte [u.quant]
212 <1> clk_2:
213 0000F371 C3 <1> retn ; return to (hardware) timer interrupt routine
214 <1>
215 <1> ; 15/01/2017
216 <1> ; 14/01/2017
217 <1> ; 07/01/2017
218 <1> ; 02/01/2017
219 <1> ; 17/08/2016
220 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

```



```

221 <1> int34h: ; #IOCTL# (I/O port access support for ring 3)
222 <1> ; 23/05/2016
223 <1> ; 20/06/2016
224 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
225 <1> ;
226 <1> ; INPUT ->
227 <1> ; AH = 0 -> read port (physical IO port) -byte-
228 <1> ; AH = 1 -> write port (physical IO port) -byte-
229 <1> ; AL = data byte
230 <1> ; AH = 2 -> read port (physical IO port) -word-
231 <1> ; AH = 3 -> write port (physical IO port) -word-
232 <1> ; BX = data word
233 <1> ; AH = 4 -> read port (physical IO port) -dword-
234 <1> ; AH = 5 -> write port (physical IO port) -dword-
235 <1> ; EBX = data dword
236 <1> ;
237 <1> ; DX = Port number (<= 0FFFFh)
238 <1> ;
239 <1> ; OUTPUT ->
240 <1> ; AL = data byte (in al, dx)
241 <1> ; AX = data word (in ax, dx)
242 <1> ; EAX = data dword (in eax, dx)
243 <1> ;
244 <1> ; (ECX = actual TRANSFER COUNT for string functions)
245 <1> ;
246 <1> ;
247 <1> ; Modified registers: EAX
248 <1> ;
249 <1>
250 0000F372 80FC05 <1> cmp ah, 5
251 0000F375 7739 <1> ja short int34h_5 ; invalid function !
252 <1>
253 <1> ;; 15/01/2017
254 <1> ; 14/01/2017
255 <1> ; 02/01/2017
256 <1> ;mov byte [ss:intflg], 34h ; IOCTL interrupt
257 0000F377 FB <1> sti
258 <1>
259 <1> ;sti ; enable interrupts
260 0000F378 80642408FE <1> and byte [esp+8], 11111110b ; clear carry bit of eflags register
261 <1>
262 0000F37D 80FC01 <1> cmp ah, 1
263 0000F380 7205 <1> jnb short int34h_0
264 0000F382 7705 <1> ja short int34h_1
265 <1>
266 0000F384 EE <1> out dx, al
267 <1> ;iretd
268 0000F385 EB01 <1> jmp short int34h_iret
269 <1>
270 <1> int34h_0:
271 0000F387 EC <1> in al, dx
272 <1> ;iretd
273 <1> int34h_iret:
274 <1> ;cli ; 07/01/2017
275 <1> ;; 15/01/2017
276 <1> ;mov byte [ss:intflg], 0 ; reset
277 0000F388 CF <1> iretd
278 <1>
279 <1> int34h_1:
280 0000F389 F6C401 <1> test ah, 1
281 0000F38C 7511 <1> jnz short int34h_3 ; out
282 <1>
283 <1> ; in
284 0000F38E 80FC02 <1> cmp ah, 2
285 0000F391 7707 <1> ja short int34h_2
286 <1>
287 0000F393 6689D8 <1> mov ax, bx
288 0000F396 66ED <1> in ax, dx
289 <1> ;iretd
290 0000F398 EBEE <1> jmp short int34h_iret
291 <1>
292 <1> int34h_2:
293 <1> ; ah = 4
294 0000F39A 89D8 <1> mov eax, ebx
295 0000F39C ED <1> in eax, dx
296 <1> ;iretd
297 0000F39D EBE9 <1> jmp short int34h_iret
298 <1>
299 <1> int34h_3:
300 0000F39F 80FC03 <1> cmp ah, 3
301 0000F3A2 7707 <1> ja short int34h_4
302 <1>
303 0000F3A4 6689D8 <1> mov ax, bx
304 0000F3A7 66EF <1> out dx, ax
305 <1> ;iretd
306 0000F3A9 EBDD <1> jmp short int34h_iret
307 <1>
308 <1> int34h_4:
309 <1> ; ah = 5
310 0000F3AB 89D8 <1> mov eax, ebx
311 0000F3AD EF <1> out dx, eax
312 <1> ;iretd
313 0000F3AE EBD8 <1> jmp short int34h_iret
314 <1>
315 <1> int34h_5:
316 0000F3B0 804C240801 <1> or byte [esp+8], 1 ; set carry bit of eflags register
317 0000F3B5 CF <1> iretd
318 <1>
319 <1> INT4Ah:
320 <1> ; 24/01/2016
321 <1> ; this procedure will be called by 'RTC_INT' (in 'timer.s')
322 0000F3B6 C3 <1> retn

```

```

323 <1>
324 <1> ; u0.s
325 <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS0.INC
326 <1> ; Last Modification: 20/11/2015
327 <1>
328 <1> com2_int:
329 <1> ; 07/11/2015
330 <1> ; 24/10/2015
331 <1> ; 23/10/2015
332 <1> ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)
333 <1> ; 28/07/2014 (Retro UNIX 8086 v1)
334 <1> ; < serial port 2 interrupt handler >
335 <1> ;
336 0000F3B7 890424 <1> mov [esp], eax ; overwrite call return address
337 <1> ;push eax
338 0000F3BA 66B80900 <1> mov ax, 9
339 0000F3BE EB07 <1> jmp short comm_int
340 <1> com1_int:
341 <1> ; 07/11/2015
342 <1> ; 24/10/2015
343 0000F3C0 890424 <1> mov [esp], eax ; overwrite call return address
344 <1> ; 23/10/2015
345 <1> ;push eax
346 0000F3C3 66B80800 <1> mov ax, 8
347 <1> comm_int:
348 <1> ; 20/11/2015
349 <1> ; 18/11/2015
350 <1> ; 17/11/2015
351 <1> ; 16/11/2015
352 <1> ; 09/11/2015
353 <1> ; 08/11/2015
354 <1> ; 07/11/2015
355 <1> ; 06/11/2015 (serial4.asm, 'serial')
356 <1> ; 01/11/2015
357 <1> ; 26/10/2015
358 <1> ; 23/10/2015
359 0000F3C7 53 <1> push ebx
360 0000F3C8 56 <1> push esi
361 0000F3C9 57 <1> push edi
362 0000F3CA 1E <1> push ds
363 0000F3CB 06 <1> push es
364 <1> ; 18/11/2015
365 0000F3CC 0F20DB <1> mov ebx, cr3
366 0000F3CF 53 <1> push ebx ; ****
367 <1> ;
368 0000F3D0 51 <1> push ecx ; ***
369 0000F3D1 52 <1> push edx ; **
370 <1> ;
371 0000F3D2 BB10000000 <1> mov ebx, KDATA
372 0000F3D7 8EDB <1> mov ds, bx
373 0000F3D9 8EC3 <1> mov es, bx
374 <1> ;
375 0000F3DB 8B0D[20380100] <1> mov ecx, [k_page_dir]
376 0000F3E1 0F22D9 <1> mov cr3, ecx
377 <1> ; 20/11/2015
378 <1> ; Interrupt identification register
379 0000F3E4 66BAFA02 <1> mov dx, 2FAh ; COM2
380 <1> ;
381 0000F3E8 3C08 <1> cmp al, 8
382 0000F3EA 7702 <1> ja short com_i0
383 <1> ;
384 <1> ; 20/11/2015
385 <1> ; 17/11/2015
386 <1> ; 16/11/2015
387 <1> ; 15/11/2015
388 <1> ; 24/10/2015
389 <1> ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)
390 <1> ; 28/07/2014 (Retro UNIX 8086 v1)
391 <1> ; < serial port 1 interrupt handler >
392 <1> ;
393 0000F3EC FEC6 <1> inc dh ; 3FAh ; COM1 Interrupt id. register
394 <1> com_i0:
395 <1> ;push eax ; *
396 <1> ; 07/11/2015
397 0000F3EE A2[8A380100] <1> mov byte [ccomport], al
398 <1> ; 09/11/2015
399 0000F3F3 0FB7D8 <1> movzx ebx, ax ; 8 or 9
400 <1> ; 17/11/2015
401 <1> ; reset request for response status
402 0000F3F6 88A3[80380100] <1> mov [ebx+req_resp-8], ah ; 0
403 <1> ;
404 <1> ; 20/11/2015
405 0000F3FC EC <1> in al, dx ; read interrupt id. register
406 0000F3FD EB00 <1> JMP $+2 ; I/O DELAY
407 0000F3FF 2404 <1> and al, 4 ; received data available?
408 0000F401 7470 <1> jz short com_eoi; (transmit. holding reg. empty)
409 <1> ;
410 <1> ; 20/11/2015
411 0000F403 80EA02 <1> sub dl, 3FAh-3F8h; data register (3F8h, 2F8h)
412 0000F406 EC <1> in al, dx ; read character
413 <1> ;JMP $+2 ; I/O DELAY
414 <1> ; 08/11/2015
415 <1> ; 07/11/2015
416 0000F407 89DE <1> mov esi, ebx
417 0000F409 89DF <1> mov edi, ebx
418 0000F40B 81C6[84380100] <1> add esi, rchar - 8 ; points to last received char
419 0000F411 81C7[86380100] <1> add edi, schar - 8 ; points to last sent char
420 0000F417 8806 <1> mov [esi], al ; received char (current char)
421 <1> ; query
422 0000F419 20C0 <1> and al, al
423 0000F41B 7527 <1> jnz short com_i2
424 <1> ; response

```

```

425      <1>      ; 17/11/2015
426      <1>      ; set request for response status
427 0000F41D FE83[80380100] <1>      inc      byte [ebx+req_resp-8] ; 1
428      <1>      ;
429 0000F423 6683C205 <1>      add     dx, 3FDh-3F8h; (3FDh, 2FDh)
430 0000F427 EC <1>      in      al, dx      ; read line status register
431 0000F428 EB00 <1>      JMP     $+2      ; I/O DELAY
432 0000F42A 2420 <1>      and     al, 20h      ; transmitter holding reg. empty?
433 0000F42C 7445 <1>      jz      short com_eoi      ; no
434 0000F42E B0FF <1>      mov     al, 0FFh      ; response
435 0000F430 6683EA05 <1>      sub     dx, 3FDh-3F8h      ; data port (3F8h, 2F8h)
436 0000F434 EE <1>      out     dx, al      ; send on serial port
437      <1>      ; 17/11/2015
438 0000F435 803F00 <1>      cmp     byte [edi], 0      ; query ? (schar)
439 0000F438 7502 <1>      jne     short com_i1      ; no
440 0000F43A 8807 <1>      mov     [edi], al      ; 0FFh (responded)
441      <1> com_i1:
442      <1>      ; 17/11/2015
443      <1>      ; reset request for response status (again)
444 0000F43C FE8B[80380100] <1>      dec     byte [ebx+req_resp-8] ; 0
445 0000F442 EB2F <1>      jmp     short com_eoi
446      <1> com_i2:
447      <1>      ; 08/11/2015
448 0000F444 3CFF <1>      cmp     al, 0FFh      ; (response ?)
449 0000F446 7417 <1>      je      short com_i3 ; (check for response signal)
450      <1>      ; 07/11/2015
451 0000F448 3C04 <1>      cmp     al, 04h      ; EOT
452 0000F44A 751C <1>      jne     short com_i4
453      <1>      ; EOT = 04h (End of Transmit) - 'CTRL + D'
454      <1>      ; (an EOT char is supposed as a ctrl+brk from the terminal)
455      <1>      ; 08/11/2015
456      <1>      ; pty -> tty 0 to 7 (pseudo screens)
457 0000F44C 861D[4E380100] <1>      xchg     bl, [ptty] ; tty number (8 or 9)
458 0000F452 E8DF6EFFFF <1>      call    ctrlbrk
459 0000F457 861D[4E380100] <1>      xchg     [ptty], bl ; (restore pty value and BL value)
460      <1>      ;mov     al, 04h ; EOT
461      <1>      ; 08/11/2015
462 0000F45D EB09 <1>      jmp     short com_i4
463      <1> com_i3:
464      <1>      ; 08/11/2015
465      <1>      ; If 0FFh has been received just after a query
466      <1>      ; (schar, ZERO), it is a response signal.
467      <1>      ; 17/11/2015
468 0000F45F 803F00 <1>      cmp     byte [edi], 0 ; query ? (schar)
469 0000F462 7704 <1>      ja      short com_i4 ; no
470      <1>      ; reset query status (schar)
471 0000F464 8807 <1>      mov     [edi], al ; 0FFh
472 0000F466 FEC0 <1>      inc     al ; 0
473      <1> com_i4:
474      <1>      ; 27/07/2014
475      <1>      ; 09/07/2014
476 0000F468 D0E3 <1>      shl     bl, 1
477 0000F46A 81C3[50380100] <1>      add     ebx, ttychr
478      <1>      ; 23/07/2014 (always overwrite)
479      <1>      ; ;cmp word [ebx], 0
480      <1>      ; ;ja short com_eoi
481      <1>      ;
482 0000F470 668903 <1>      mov     [ebx], ax      ; Save ascii code
483      <1>      ; scan code = 0
484      <1> com_eoi:
485      <1>      ;mov     al, 20h
486      <1>      ;out     20h, al      ; end of interrupt
487      <1>      ;
488      <1>      ; 07/11/2015
489      <1>      ;pop     eax ; *
490 0000F473 A0[8A380100] <1>      mov     al, byte [ccomport] ; current COM port
491      <1>      ; al = tty number (8 or 9)
492 0000F478 E85E010000 <1>      call    wakeup
493      <1> com_i2ret:
494      <1>      ; 23/10/2015
495 0000F47D 5A <1>      pop     edx ; **
496 0000F47E 59 <1>      pop     ecx ; ***
497      <1>      ; 18/11/2015
498      <1>      ;pop     eax ; ****
499      <1>      ;mov     cr3, eax
500      <1>      ;jmp     iiret
501 0000F47F E95C16FFFF <1>      jmp     iiretp
502      <1>
503      <1> ;iiretp: ; 01/09/2015
504      <1> ;      ; 28/08/2015
505      <1> ;      pop     eax ; (*) page directory
506      <1> ;      mov     cr3, eax
507      <1> ;iiret:
508      <1> ;      ; 22/08/2014
509      <1> ;      mov     al, 20h ; END OF INTERRUPT COMMAND TO 8259
510      <1> ;      out     20h, al      ; 8259 PORT
511      <1> ;      ;
512      <1> ;      pop     es
513      <1> ;      pop     ds
514      <1> ;      pop     edi
515      <1> ;      pop     esi
516      <1> ;      pop     ebx ; 29/08/2014
517      <1> ;      pop     eax
518      <1> ;      iretd
519      <1>
520      <1> sp_init:
521      <1>      ; 07/11/2015
522      <1>      ; 29/10/2015
523      <1>      ; 26/10/2015
524      <1>      ; 23/10/2015
525      <1>      ; 29/06/2015
526      <1>      ; 14/03/2015 (Retro UNIX 386 v1 - 115200 baud)

```

```

527 <1> ; 28/07/2014 (Retro UNIX 8086 v1 - 9600 baud)
528 <1> ; Initialization of Serial Port Communication Parameters
529 <1> ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
530 <1> ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
531 <1> ;
532 <1> ; ((Modified registers: EAX, ECX, EDX, EBX))
533 <1> ;
534 <1> ; INPUT: (29/06/2015)
535 <1> ; AL = 0 for COM1
536 <1> ; 1 for COM2
537 <1> ; AH = Communication parameters
538 <1> ;
539 <1> ; (*) Communication parameters (except BAUD RATE):
540 <1> ; Bit 4 3 2 1 0
541 <1> ; -PARITY-- STOP BIT -WORD LENGTH-
542 <1> ; this one --> 00 = none 0 = 1 bit 11 = 8 bits
543 <1> ; 01 = odd 1 = 2 bits 10 = 7 bits
544 <1> ; 11 = even
545 <1> ; Baud rate setting bits: (29/06/2015)
546 <1> ; Retro UNIX 386 v1 feature only !
547 <1> ; Bit 7 6 5 | Baud rate
548 <1> ; -----
549 <1> ; value 0 0 0 | Default (Divisor = 1)
550 <1> ; 0 0 1 | 9600 (12)
551 <1> ; 0 1 0 | 19200 (6)
552 <1> ; 0 1 1 | 38400 (3)
553 <1> ; 1 0 0 | 14400 (8)
554 <1> ; 1 0 1 | 28800 (4)
555 <1> ; 1 1 0 | 57600 (2)
556 <1> ; 1 1 1 | 115200 (1)
557 <1>
558 <1> ; References:
559 <1> ; (1) IBM PC-XT Model 286 BIOS Source Code
560 <1> ; RS232.ASM --- 10/06/1985 COMMUNICATIONS BIOS (RS232)
561 <1> ; (2) Award BIOS 1999 - ATORG.S.ASM
562 <1> ; (3) http://wiki.osdev.org/Serial_Ports
563 <1> ;
564 <1> ; Set communication parameters for COM1 (= 03h)
565 <1> ;
566 0000F484 BB[86380100] <1> mov ebx, compl ; COM1 parameters
567 0000F489 66BAF803 <1> mov dx, 3F8h ; COM1
568 <1> ; 29/10/2015
569 0000F48D 66B90103 <1> mov cx, 301h ; divisor = 1 (115200 baud)
570 0000F491 E86F000000 <1> call sp_i3 ; call A4
571 0000F496 A880 <1> test al, 80h
572 0000F498 7410 <1> jz short sp_i0 ; OK..
573 <1> ; Error !
574 <1> ;mov dx, 3F8h
575 0000F49A 80EA05 <1> sub dl, 5 ; 3FDh -> 3F8h
576 0000F49D 66B90E03 <1> mov cx, 30Eh ; divisor = 12 (9600 baud)
577 0000F4A1 E85F000000 <1> call sp_i3 ; call A4
578 0000F4A6 A880 <1> test al, 80h
579 0000F4A8 7508 <1> jnz short sp_i1
580 <1> sp_i0:
581 <1> ; (Note: Serial port interrupts will be disabled here...)
582 <1> ; (INT 14h initialization code disables interrupts.)
583 <1> ;
584 0000F4AA C603E3 <1> mov byte [ebx], 0E3h ; 11100011b
585 0000F4AD E8DC000000 <1> call sp_i5 ; 29/06/2015
586 <1> sp_i1:
587 0000F4B2 43 <1> inc ebx
588 0000F4B3 66BAF802 <1> mov dx, 2F8h ; COM2
589 <1> ; 29/10/2015
590 0000F4B7 66B90103 <1> mov cx, 301h ; divisor = 1 (115200 baud)
591 0000F4BB E845000000 <1> call sp_i3 ; call A4
592 0000F4C0 A880 <1> test al, 80h
593 0000F4C2 7410 <1> jz short sp_i2 ; OK..
594 <1> ; Error !
595 <1> ;mov dx, 2F8h
596 0000F4C4 80EA05 <1> sub dl, 5 ; 2FDh -> 2F8h
597 0000F4C7 66B90E03 <1> mov cx, 30Eh ; divisor = 12 (9600 baud)
598 0000F4CB E835000000 <1> call sp_i3 ; call A4
599 0000F4D0 A880 <1> test al, 80h
600 0000F4D2 7530 <1> jnz short sp_i7
601 <1> sp_i2:
602 0000F4D4 C603E3 <1> mov byte [ebx], 0E3h ; 11100011b
603 <1> sp_i6:
604 <1> ;; COM2 - enabling IRQ 3
605 <1> ; 07/11/2015
606 <1> ; 26/10/2015
607 0000F4D7 9C <1> pushf
608 0000F4D8 FA <1> cli
609 <1> ;
610 0000F4D9 66BAFC02 <1> mov dx, 2FCh ; modem control register
611 0000F4DD EC <1> in al, dx ; read register
612 0000F4DE EB00 <1> JMP $+2 ; I/O DELAY
613 0000F4E0 0C08 <1> or al, 8 ; enable bit 3 (OUT2)
614 0000F4E2 EE <1> out dx, al ; write back to register
615 0000F4E3 EB00 <1> JMP $+2 ; I/O DELAY
616 0000F4E5 66BAF902 <1> mov dx, 2F9h ; interrupt enable register
617 0000F4E9 EC <1> in al, dx ; read register
618 0000F4EA EB00 <1> JMP $+2 ; I/O DELAY
619 <1> ;or al, 1 ; receiver data interrupt enable and
620 0000F4EC 0C03 <1> or al, 3 ; transmitter empty interrupt enable
621 0000F4EE EE <1> out dx, al ; write back to register
622 0000F4EF EB00 <1> JMP $+2 ; I/O DELAY
623 0000F4F1 E421 <1> in al, 21h ; read interrupt mask register
624 0000F4F3 EB00 <1> JMP $+2 ; I/O DELAY
625 0000F4F5 24F7 <1> and al, 0F7h ; enable IRQ 3 (COM2)
626 0000F4F7 E621 <1> out 21h, al ; write back to register
627 <1> ;
628 <1> ; 23/10/2015

```


444

```

629 0000F4F9 B8[B7F30000]      <1>          mov     eax, com2_int
630 0000F4FE A3[D6F50000]      <1>          mov     [com2_irq3], eax
631                                <1>          ; 26/10/2015
632 0000F503 9D                <1>          popf
633                                <1> sp_i7:
634 0000F504 C3                <1>          retn
635                                <1>
636                                <1> sp_i3:
637                                <1> ;A4:           ;----- INITIALIZE THE COMMUNICATIONS PORT
638                                <1>           ; 28/10/2015
639 0000F505 FEC2              <1>          inc     dl       ; 3F9h (2F9h); 3F9h, COM1 Interrupt enable register
640 0000F507 B000              <1>          mov     al, 0
641 0000F509 EE                <1>          out     dx, al         ; disable serial port interrupt
642 0000F50A EB00              <1>          JMP     $+2             ; I/O DELAY
643 0000F50C 80C202            <1>          add     dl, 2   ; 3FBh (2FBh); COM1 Line control register (3FBh)
644 0000F50F B080              <1>          mov     al, 80h
645 0000F511 EE                <1>          out     dx, al         ; SET DLAB=1 ; divisor latch access bit
646                                <1>           ;----- SET BAUD RATE DIVISOR
647                                <1>           ; 26/10/2015
648 0000F512 80EA03            <1>          sub     dl, 3    ; 3F8h (2F8h)        ; register for least significant byte
649                                <1>                                     ; of the divisor value
650 0000F515 88C8              <1>          mov     al, cl ; 1
651 0000F517 EE                <1>          out     dx, al         ; 1 = 115200 baud (Retro UNIX 386 v1)
652                                <1>                                     ; 2 = 57600 baud
653                                <1>                                     ; 3 = 38400 baud
654                                <1>                                     ; 6 = 19200 baud
655                                <1>                                     ; 12 = 9600 baud (Retro UNIX 8086 v1)
656 0000F518 EB00              <1>          JMP     $+2             ; I/O DELAY
657 0000F51A 28C0              <1>          sub     al, al
658 0000F51C FEC2              <1>          inc     dl       ; 3F9h (2F9h)        ; register for most significant byte
659                                <1>                                     ; of the divisor value
660 0000F51E EE                <1>          out     dx, al ; 0
661 0000F51F EB00              <1>          JMP     $+2             ; I/O DELAY
662                                <1>          ;
663 0000F521 88E8              <1>          mov     al, ch ; 3          ; 8 data bits, 1 stop bit, no parity
664                                <1>          ;and al, 1Fh ; Bits 0,1,2,3,4
665 0000F523 80C202            <1>          add     dl, 2   ; 3FBh (2FBh); Line control register
666 0000F526 EE                <1>          out     dx, al
667 0000F527 EB00              <1>          JMP     $+2             ; I/O DELAY
668                                <1>           ; 29/10/2015
669 0000F529 FECA              <1>          dec     dl       ; 3FAh (2FAh); FIFO Control register (16550/16750)
670 0000F52B 30C0              <1>          xor     al, al         ; 0
671 0000F52D EE                <1>          out     dx, al         ; Disable FIFOs (reset to 8250 mode)
672 0000F52E EB00              <1>          JMP     $+2
673                                <1> sp_i4:
674                                <1> ;A18:           ;----- COMM PORT STATUS ROUTINE
675                                <1>           ; 29/06/2015 (line status after modem status)
676 0000F530 80C204            <1>          add     dl, 4   ; 3FEh (2FEh); Modem status register
677                                <1> sp_i4s:
678 0000F533 EC                <1>          in      al, dx         ; GET MODEM CONTROL STATUS
679 0000F534 EB00              <1>          JMP     $+2             ; I/O DELAY
680 0000F536 88C4              <1>          mov     ah, al         ; PUT IN (AH) FOR RETURN
681 0000F538 FECA              <1>          dec     dl       ; 3FDh (2FDh); POINT TO LINE STATUS REGISTER
682                                <1>                                     ; dx = 3FDh for COM1, 2FDh for COM2
683 0000F53A EC                <1>          in      al, dx         ; GET LINE CONTROL STATUS
684                                <1>          ; AL = Line status, AH = Modem status
685 0000F53B C3                <1>          retn
686                                <1>
687                                <1> sp_status:
688                                <1>           ; 29/06/2015
689                                <1>           ; 27/06/2015 (Retro UNIX 386 v1)
690                                <1>           ; Get serial port status
691 0000F53C 66BAFE03            <1>          mov     dx, 3FEh        ; Modem status register (COM1)
692 0000F540 28C6              <1>          sub     dh, al         ; dh = 2 for COM2 (al = 1)
693                                <1>                                     ; dx = 2FEh for COM2
694 0000F542 EBEF              <1>          jmp     short sp_i4s
695                                <1>
696                                <1> sp_setp: ; Set serial port communication parameters
697                                <1>           ; 07/11/2015
698                                <1>           ; 29/10/2015
699                                <1>           ; 29/06/2015
700                                <1>           ; Retro UNIX 386 v1 feature only !
701                                <1>           ;
702                                <1>           ; INPUT:
703                                <1>           ;     AL = 0 for COM1
704                                <1>           ;     1 for COM2
705                                <1>           ;     AH = Communication parameters (*)
706                                <1>           ; OUTPUT:
707                                <1>           ;     CL = Line status
708                                <1>           ;     CH = Modem status
709                                <1>           ; If cf = 1 -> Error code in [u.error]
710                                <1>           ;               'invalid parameter !'
711                                <1>           ;               or
712                                <1>           ;               'device not ready !' error
713                                <1>           ;
714                                <1>           ; (*) Communication parameters (except BAUD RATE):
715                                <1>           ; Bit      4      3      2      1      0
716                                <1>           ;           -PARITY-- STOP BIT -WORD LENGTH-
717                                <1>           ; this one --> 00 = none    0 = 1 bit  11 = 8 bits
718                                <1>           ;           01 = odd     1 = 2 bits  10 = 7 bits
719                                <1>           ;           11 = even
720                                <1>           ; Baud rate setting bits: (29/06/2015)
721                                <1>           ; Retro UNIX 386 v1 feature only !
722                                <1>           ; Bit      7      6      5 | Baud rate
723                                <1>           ; -----
724                                <1>           ; value 0      0      0 | Default (Divisor = 1)
725                                <1>           ;           0      0      1 | 9600 (12)
726                                <1>           ;           0      1      0 | 19200 (6)
727                                <1>           ;           0      1      1 | 38400 (3)
728                                <1>           ;           1      0      0 | 14400 (8)
729                                <1>           ;           1      0      1 | 28800 (4)
730                                <1>           ;           1      1      0 | 57600 (2)

```

```

731      <1>      ;          1    1    1    | 115200 (1)
732      <1>      ;
733      <1>      ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
734      <1>      ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
735      <1>      ;
736      <1>      ; ((Modified registers: EAX, ECX, EDX, EBX))
737      <1>      ;
738 0000F544 66BAF803      <1>      mov     dx, 3F8h
739 0000F548 BB[86380100]  <1>      mov     ebx, comlp ; COM1 control byte offset
740 0000F54D 3C01          <1>      cmp     al, 1
741 0000F54F 776B          <1>      ja      short sp_invp_err
742 0000F551 7203          <1>      jb      short sp_setpl ; COM1 (AL = 0)
743 0000F553 FECE          <1>      dec     dh ; 2F8h
744 0000F555 43           <1>      inc     ebx ; COM2 control byte offset
745      <1>      sp_setpl:
746      <1>      ; 29/10/2015
747 0000F556 8823          <1>      mov     [ebx], ah
748 0000F558 0FB6CC          <1>      movzx   ecx, ah
749 0000F55B C0E905          <1>      shr     cl, 5 ; -> baud rate index
750 0000F55E 80E41F          <1>      and     ah, 1Fh ; communication parameters except baud rate
751 0000F561 8A81[CBF50000]  <1>      mov     al, [ecx+b_div_tbl]
752 0000F567 6689C1          <1>      mov     cx, ax
753 0000F56A E896FFFFFF          <1>      call    sp_i3
754 0000F56F 6689C1          <1>      mov     cx, ax ; CL = Line status, CH = Modem status
755 0000F572 A880          <1>      test    al, 80h
756 0000F574 740F          <1>      jz      short sp_setp2
757 0000F576 C603E3          <1>      mov     byte [ebx], 0E3h ; Reset to initial value (11100011b)
758      <1>      stp_dnr_err:
759 0000F579 C705[104F0100]0F00- <1>      mov     dword [u.error], ERR_DEV_NOT_RDY ; 'device not ready !'
759 0000F581 0000          <1>
760      <1>      ; CL = Line status, CH = Modem status
761 0000F583 F9           <1>      stc
762 0000F584 C3           <1>      retn
763      <1>      sp_setp2:
764 0000F585 80FE02          <1>      cmp     dh, 2 ; COM2 (2F?h)
765 0000F588 0F8649FFFFFF          <1>      jna      sp_i6
766      <1>      ; COM1 (3F?h)
767      <1>      sp_i5:
768      <1>      ; 07/11/2015
769      <1>      ; 26/10/2015
770      <1>      ; 29/06/2015
771      <1>      ;
772      <1>      ;; COM1 - enabling IRQ 4
773 0000F58E 9C           <1>      pushf
774 0000F58F FA           <1>      cli
775 0000F590 66BAFC03      <1>      mov     dx, 3FCh ; modem control register
776 0000F594 EC           <1>      in      al, dx ; read register
777 0000F595 EB00          <1>      JMP     $+2 ; I/O DELAY
778 0000F597 0C08          <1>      or      al, 8 ; enable bit 3 (OUT2)
779 0000F599 EE           <1>      out     dx, al ; write back to register
780 0000F59A EB00          <1>      JMP     $+2 ; I/O DELAY
781 0000F59C 66BAF903      <1>      mov     dx, 3F9h ; interrupt enable register
782 0000F5A0 EC           <1>      in      al, dx ; read register
783 0000F5A1 EB00          <1>      JMP     $+2 ; I/O DELAY
784      <1>      ;or al, 1 ; receiver data interrupt enable and
785 0000F5A3 0C03          <1>      or      al, 3 ; transmitter empty interrupt enable
786 0000F5A5 EE           <1>      out     dx, al ; write back to register
787 0000F5A6 EB00          <1>      JMP     $+2 ; I/O DELAY
788 0000F5A8 E421          <1>      in      al, 21h ; read interrupt mask register
789 0000F5AA EB00          <1>      JMP     $+2 ; I/O DELAY
790 0000F5AC 24EF          <1>      and     al, 0EFh ; enable IRQ 4 (COM1)
791 0000F5AE E621          <1>      out     21h, al ; write back to register
792      <1>      ;
793      <1>      ; 23/10/2015
794 0000F5B0 B8[C0F30000]  <1>      mov     eax, com1_int
795 0000F5B5 A3[D2F50000]  <1>      mov     [com1_irq4], eax
796      <1>      ; 26/10/2015
797 0000F5BA 9D           <1>      popf
798 0000F5BB C3           <1>      retn
799      <1>
800      <1>      sp_invp_err:
801 0000F5BC C705[104F0100]1700- <1>      mov     dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
801 0000F5C4 0000          <1>
802 0000F5C6 31C9          <1>      xor     ecx, ecx
803 0000F5C8 49           <1>      dec     ecx ; 0FFFFh
804 0000F5C9 F9           <1>      stc
805 0000F5CA C3           <1>      retn
806      <1>
807      <1>      ; 29/10/2015
808      <1>      b_div_tbl: ; Baud rate divisor table (115200/divisor)
809 0000F5CB 010C0603080401 <1>      db 1, 12, 6, 3, 8, 4, 1
810      <1>
811      <1>
812      <1>      ; 23/10/2015
813      <1>      com1_irq4:
814 0000F5D2 [DAF50000]  <1>      dd dummy_retn
815      <1>      com2_irq3:
816 0000F5D6 [DAF50000]  <1>      dd dummy_retn
817      <1>
818      <1>      dummy_retn:
819 0000F5DA C3           <1>      retn
820      <1>
821      <1>      wakeup:
822      <1>      ; 24/01/2016
823 0000F5DB C3           <1>      retn
824      <1>
825      <1>      set_working_path_x:
826      <1>      ; 17/10/2016 (TRDOS 386 - FFF & FNF)
827 0000F5DC 66B80100      <1>      mov     ax, 1
828      <1>      ; File name is needed/forced (AL=1)
829      <1>      ; Change directory as temporary (AH=0)
830      <1>

```

```

831          <1>          ; This is needed for preventing wrong Find Next File
832          <1>          ; system call after sysopen, syscreate, sysmkdir etc.
833          <1>          ; Find Next File must immediate follow Find First file)
834          <1>
835 0000F5E0 8825[D8450100] <1>          mov     [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
836          <1>
837          <1> set_working_path:
838          <1>          ; 16/10/2016
839          <1>          ; 12/10/2016
840          <1>          ; 10/10/2016
841          <1>          ; 05/10/2016 - TRDOS 386 (TRDOS v2.0)
842          <1>          ;
843          <1>          ; TRDOS v1.0 (DIR.ASM, "proc_set_working_path")
844          <1>          ; 27/01/2011 - 08/02/2011
845          <1>          ; Set/Changes current drive, directory and file
846          <1>          ; depending on command tail
847          <1>          ; (procedure is derivated from CMD_INTR.ASM
848          <1>          ; file or dir locating code of internal commands)
849          <1>          ; (This procedure is prepared for INT 21H file/dir
850          <1>          ; functions and also to get compact code for
851          <1>          ; internal mainprog -command interpreter- commands)
852          <1>          ;
853          <1>          ; INPUT: DS:SI -> Command tail (ASCIIIZ string)
854          <1>          ; AL= 0 -> any, AL > 0 -> file name is forced
855          <1>          ; AH= CD -> Change directory permanently
856          <1>          ; AH <> CD -> Change directory as temporary
857          <1>          ;
858          <1>          ; OUTPUT: ES=DS, FindFile structure has been set
859          <1>          ;         RUN_CDRV points previous current drive
860          <1>          ;         DS:SI = FindFile structure address
861          <1>          ;         (DS=CS)
862          <1>          ;         AX, BX, CX, DX, DI will be changed
863          <1>          ;         cf = 1 -> Error code in AX (AL)
864          <1>          ;         stc & AX = 0 -> Bad command or path name
865          <1>          ; -----
866          <1>          ;
867          <1>          ; TRDOS 386 (05/10/2016)
868          <1>          ; INPUT:
869          <1>          ;         ESI = File/Directory Path (ASCIIIZ string)
870          <1>          ;         address in user's memory space
871          <1>          ;         Al = 0 -> any
872          <1>          ;         AL > 0 -> file name is forced
873          <1>          ;         AH = CD -> change directory as permanent
874          <1>          ;         AH <> CD -> change directory as temporary
875          <1>          ;
876          <1>          ; OUTPUT:
877          <1>          ;         FindFile structure has been set
878          <1>          ;         RUN_CDRV points previous current drive
879          <1>          ;         ESI = FindFile_Name address ; 12/10/2016
880          <1>          ;
881          <1>          ;         cf = 1 -> Error code in EAX (AL)
882          <1>          ;         stc & EAX = 0 -> Bad command or path name
883          <1>          ;
884          <1>          ; Modified registers: EAX, EBX, ECX, EDX, ESI, EDI
885          <1>
886 0000F5E6 66A3[DC450100] <1>          mov     [SWP_Model], ax
887 0000F5EC A0[E6380100] <1>          mov     al, [Current_Drv]
888 0000F5F1 30E4 <1>          xor     ah, ah
889 0000F5F3 66A3[DE450100] <1>          mov     word [SWP_DRV], ax
890          <1>
891          <1>          ; TRDOS 386 ring 3 (user's page directory)
892          <1>          ; to ring 0 (kernel's page directory)
893          <1>          ; transfer modifications (05/10/2016).
894          <1>
895 0000F5F9 55 <1>          push    ebp
896 0000F5FA 89E5 <1>          mov     ebp, esp
897          <1>
898 0000F5FC B980000000 <1>          mov     ecx, 128 ; maximum path length = 128 bytes
899 0000F601 29CC <1>          sub     esp, ecx ; reserve 128 bytes (buffer) on stack
900 0000F603 89E7 <1>          mov     edi, esp ; destination address (kernel space)
901          <1>          ; esi = source address (virtual, in user's memory space)
902 0000F605 E810F2FFFF <1>          call    transfer_from_user_buffer
903 0000F60A 720A <1>          jc     short loc_swp_xor_retn
904          <1>
905 0000F60C 89E6 <1>          mov     esi, esp ; temporary buffer (the path) on stack
906          <1> loc_swp_fchar:
907 0000F60E 8A06 <1>          mov     al, [esi]
908 0000F610 3C20 <1>          cmp     al, 20h
909 0000F612 7711 <1>          ja     short loc_swp_parse_path_name
910 0000F614 740C <1>          je     short loc_swp_fchar_next
911          <1>
912          <1> loc_swp_xor_retn:
913 0000F616 31C0 <1>          xor     eax, eax
914 0000F618 F9 <1>          stc
915          <1> loc_swp_retn:
916 0000F619 89EC <1>          mov     esp, ebp
917 0000F61B 5D <1>          pop     ebp
918          <1>
919          <1>          ;mov     esi, FindFile_Drv
920 0000F61C BE[CC420100] <1>          mov     esi, FindFile_Name ; 12/10/2016
921 0000F621 C3 <1>          retn
922          <1>
923          <1> loc_swp_fchar_next:
924 0000F622 46 <1>          inc     esi
925 0000F623 EBE9 <1>          jmp     short loc_swp_fchar
926          <1>
927          <1> loc_swp_parse_path_name:
928 0000F625 BF[8A420100] <1>          mov     edi, FindFile_Drv
929 0000F62A E837ABFFFF <1>          call    parse_path_name
930 0000F62F 72E8 <1>          jc     short loc_swp_retn
931          <1>
932          <1> loc_swp_checkfile_name:

```

```

933 0000F631 803D[DC450100]00 <1>      cmp     byte [SWP_Model], 0
934 0000F638 761E <1>      jna     short loc_swp_drv
935 <1>
936 <1>      ; 10/10/2016 (valid file name checking)
937 0000F63A BE[CC420100] <1>      mov     esi, FindFile_Name
938 0000F63F 803E20 <1>      cmp     byte [esi], 20h
939 0000F642 76D2 <1>      jna     short loc_swp_xor_retn
940 <1>
941 <1>      ; 16/10/2016
942 0000F644 C605[DB450100]00 <1>      mov     byte [SWP_inv_fname], 0 ; reset
943 <1>      ; esi = file name address (ASCIIIZ)
944 0000F64B E8BF8CFFFF <1>      call    check_filename
945 0000F650 7306 <1>      jnc     short loc_swp_drv
946 <1>
947 0000F652 FE05[DB450100] <1>      inc     byte [SWP_inv_fname] ; set
948 <1> loc_swp_drv:
949 0000F658 8A35[E6380100] <1>      mov     dh, [Current_Drv]
950 <1>      ;mov     [RUN_CDRV], dh
951 <1>
952 0000F65E 8A15[8A420100] <1>      mov     dl, [FindFile_Drv]
953 <1>      ;cmp     dl, dh
954 0000F664 3A15[E6380100] <1>      cmp     dl, [Current_Drv]
955 0000F66A 740D <1>      je      short loc_swp_change_directory
956 <1>
957 0000F66C FE05[DF450100] <1>      inc     byte [SWP_DRV_chg]
958 0000F672 E84675FFFF <1>      call    change_current_drive
959 0000F677 72A0 <1>      jc      short loc_swp_retn ; eax = error code
960 <1>      ; eax = 0
961 <1>
962 <1> loc_swp_change_directory:
963 0000F679 803D[8B420100]21 <1>      cmp     byte [FindFile_Directory], 21h
964 0000F680 F5 <1>      cmc
965 0000F681 7396 <1>      jnc     short loc_swp_retn
966 <1>
967 0000F683 FE05[DF450100] <1>      inc     byte [SWP_DRV_chg]
968 0000F689 FE05[A8FA0000] <1>      inc     byte [Restore_CDIR]
969 0000F68F BE[8B420100] <1>      mov     esi, FindFile_Directory
970 0000F694 8A25[DD450100] <1>      mov     ah, [SWP_Mode+1]
971 0000F69A E8B3A4FFFF <1>      call    change_current_directory
972 0000F69F 0F8274FFFFFF <1>      jc      loc_swp_retn ; eax = error code
973 <1>
974 <1> loc_swp_change_prompt_dir_string:
975 <1>      ; esi = PATH_Array
976 <1>      ; eax = Current Directory First Cluster
977 <1>      ; edi = Logical DOS Drive Description Table
978 0000F6A5 E8CDA3FFFF <1>      call    change_prompt_dir_str
979 0000F6AA 29C0 <1>      sub     eax, eax ; 0
980 0000F6AC E968FFFFFF <1>      jmp     loc_swp_retn
981 <1>
982 <1> reset_working_path:
983 <1>      ; 06/10/2016 - TRDOS 386 (TRDOS v2.0)
984 <1>      ;
985 <1>      ; TRDOS v1.0 (DIR.ASM, "proc_reset_working_path")
986 <1>      ; 05/02/2011 - 08/02/2011
987 <1>      ;
988 <1>      ; Restores current drive and directory
989 <1>      ;
990 <1>      ; INPUT: none
991 <1>      ; OUTPUT: DL = SWP_DRV, EAX = 0 -> OK
992 <1>      ;
993 <1>      ; AX = 0 -> ESI = Logical Dos Drv Desc. Table
994 <1>      ;
995 <1>      ; EAX, EBX, ECX, EDX, ESI, EDI will be changed
996 <1>      ;
997 <1>
998 <1>
999 0000F6B1 31C0 <1>      xor     eax, eax
1000 0000F6B3 48 <1>      dec     eax
1001 <1>
1002 0000F6B4 668B15[DE450100] <1>      mov     dx, [SWP_DRV]
1003 0000F6BB 08F6 <1>      or      dh, dh
1004 0000F6BD 742E <1>      jz      short loc_rwp_return
1005 <1>
1006 0000F6BF 3A15[E6380100] <1>      cmp     dl, [Current_Drv]
1007 0000F6C5 7407 <1>      je      short loc_rwp_restore_cdir
1008 <1> loc_rwp_restore_cdrv:
1009 0000F6C7 E8F174FFFF <1>      call    change_current_drive
1010 0000F6CC EB10 <1>      jmp     short loc_rwp_restore_ok
1011 <1> loc_rwp_restore_cdir:
1012 0000F6CE 31DB <1>      xor     ebx, ebx
1013 0000F6D0 88D7 <1>      mov     bh, dl
1014 0000F6D2 BE00010900 <1>      mov     esi, Logical_DOSDisks
1015 0000F6D7 01DE <1>      add     esi, ebx
1016 <1>
1017 0000F6D9 E89675FFFF <1>      call    restore_current_directory
1018 <1>
1019 <1> loc_rwp_restore_ok:
1020 0000F6DE 668B15[DE450100] <1>      mov     dx, [SWP_DRV]
1021 0000F6E5 31C0 <1>      xor     eax, eax
1022 0000F6E7 66A3[DF450100] <1>      mov     [SWP_DRV_chg], ax
1023 <1> loc_rwp_return:
1024 0000F6ED C3 <1>      retn
1025 <1>
1026 <1> get_file_name:
1027 <1>      ; 15/10/2016 - TRDOS 386 (TRDOS v2.0)
1028 <1>      ; Convert file name
1029 <1>      ; from directory entry format
1030 <1>      ; to (8.3) dot file name format
1031 <1>      ;
1032 <1>      ; TRDOS v1.0 (DIR.ASM, "get_file_name")
1033 <1>      ; 2005 - 09/10/2011
1034 <1>      ; INPUT:

```



```

1035      <1>          ;      DS:SI -> Directory Entry Format File Name
1036      <1>          ;      ES:DI -> DOS Dot File Name Address
1037      <1>          ; OUTPUT:
1038      <1>          ;      DS:SI -> DOS Dot File Name Address
1039      <1>          ;      ES:DI -> Directory Entry Format File Name
1040      <1>          ;
1041      <1>          ; TRDOS 386 (15/10/2016)
1042      <1>          ; INPUT:
1043      <1>          ;      ESI = File name addr in dir entry format
1044      <1>          ;      EDI = Dot file name address (destination)
1045      <1>          ; OUTPUT:
1046      <1>          ;      File name is converted and moved
1047      <1>          ;      to destination (as 8.3 dot filename)
1048      <1>          ;
1049      <1>          ; Modified registers: EAX, ECX
1050      <1>
1051      <1>          ; 2005 (TRDOS 8086) - 2016 (TRDOS 386)
1052      <1>
1053      0000F6EE 57      <1>          push    edi
1054      0000F6EF 56      <1>          push    esi
1055      0000F6F0 AC      <1>          lodsb
1056      0000F6F1 3C20    <1>          cmp     al, 20h
1057      0000F6F3 762A    <1>          jna     short pass_gfn_ext
1058      0000F6F5 56      <1>          push    esi
1059      0000F6F6 AA      <1>          stosb
1060      0000F6F7 B907000000 <1>          mov     ecx, 7
1061      <1> loc_gfn_next_char:
1062      0000F6FC AC      <1>          lodsb
1063      0000F6FD 3C20    <1>          cmp     al, 20h
1064      0000F6FF 7603    <1>          jna     short pass_gfn_fn
1065      0000F701 AA      <1>          stosb
1066      0000F702 E2F8    <1>          loop    loc_gfn_next_char
1067      <1> pass_gfn_fn:
1068      0000F704 5E      <1>          pop     esi
1069      0000F705 83C607   <1>          add     esi, 7
1070      0000F708 AC      <1>          lodsb
1071      0000F709 3C20    <1>          cmp     al, 20h
1072      0000F70B 7612    <1>          jna     short pass_gfn_ext
1073      0000F70D B42E    <1>          mov     ah, '.'
1074      0000F70F 86E0    <1>          xchg    ah, al
1075      0000F711 66AB    <1>          stosw
1076      0000F713 AC      <1>          lodsb
1077      0000F714 3C20    <1>          cmp     al, 20h
1078      0000F716 7607    <1>          jna     short pass_gfn_ext
1079      0000F718 AA      <1>          stosb
1080      0000F719 AC      <1>          lodsb
1081      0000F71A 3C20    <1>          cmp     al, 20h
1082      0000F71C 7601    <1>          jna     short pass_gfn_ext
1083      0000F71E AA      <1>          stosb
1084      <1> pass_gfn_ext:
1085      0000F71F 30C0    <1>          xor     al, al
1086      0000F721 AA      <1>          stosb
1087      0000F722 5E      <1>          pop     esi
1088      0000F723 5F      <1>          pop     edi
1089      0000F724 C3      <1>          retn
1090      <1>
1091      <1>
1092      <1> set_hardware_int_vector:
1093      <1>          ; 18/03/2017
1094      <1>          ; 03/03/2017
1095      <1>          ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)
1096      <1>          ;
1097      <1>          ; SET/RESET HARDWARE INTERRUPT GATE
1098      <1>          ;
1099      <1>          ; Changes interrupt gate descriptor table
1100      <1>          ; (without changing default interrupt list)
1101      <1>          ;
1102      <1>          ; INPUT:
1103      <1>          ;      AL = IRQ number (0 to 15)
1104      <1>          ;      AH > 0 -> set
1105      <1>          ;      AH = 0 -> reset
1106      <1>          ;
1107      <1>          ; Modifed registers: eax, ebx, edx, edi
1108      <1>          ;
1109      <1>
1110      0000F725 C0E002   <1>          shl     al, 2 ; IRQ number * 4
1111      0000F728 0FB6D8   <1>          movzx    ebx, al
1112      <1>
1113      0000F72B 08E4      <1>          or      ah, ah
1114      0000F72D 7508      <1>          jnz     short shintv_1 ; set (for user call service)
1115      <1>
1116      <1>          ; 18/03/2017
1117      0000F72F 81C3[74040100] <1>          add     ebx, IRQ_list ; reset to default interrupt list
1118      0000F735 EB06      <1>          jmp     short shintv_2
1119      <1> shintv_1:
1120      0000F737 81C3[5EF70000] <1>          add     ebx, IRQ_u_list
1121      <1> shintv_2:
1122      0000F73D 8B13      <1>          mov     edx, [ebx] ; IRQ handler address
1123      <1>
1124      <1>          ; 03/03/2017
1125      0000F73F D0E0      <1>          shl     al, 1 ; IRQ number * 8
1126      <1>          ; 18/03/2017
1127      0000F741 0FB6F8   <1>          movzx    edi, al
1128      0000F744 81C7[38360100] <1>          add     edi, idt + (8*32) ; IRQ 0 offset = idt + 128
1129      <1>
1130      0000F74A 89D0      <1>          mov     eax, edx ; IRQ handler address
1131      0000F74C BB00000800 <1>          mov     ebx, 80000h
1132      <1>
1133      <1>          ;mov    edx, eax
1134      0000F751 66BA008E   <1>          mov     dx, 8E00h
1135      0000F755 6689C3     <1>          mov     bx, ax
1136      0000F758 89D8      <1>          mov     eax, ebx ; /* selector = 0x0008 = cs */

```

```

1137                                     <1>                                     ; /* interrupt gate - dpl=0, present */
1138 0000F75A AB                          <1>                                stosd  ; selector & offset bits 0-15
1139 0000F75B 8917                       <1>                                mov   [edi], edx ; attributes & offset bits 16-23
1140                                     <1>
1141 0000F75D C3                          <1>                                retn
1142                                     <1> IRQ_u_list:
1143                                     <1>                                ; 28/02/2017
1144 0000F75E [8B060000]                  <1>                                dd     timer_int
1145 0000F762 [FF0D0000]                  <1>                                dd     kb_int
1146 0000F766 [6D080000]                  <1>                                dd     irq2
1147 0000F76A [9EF70000]                  <1>                                dd     IRQ_service3
1148 0000F76E [A8F70000]                  <1>                                dd     IRQ_service4
1149 0000F772 [B2F70000]                  <1>                                dd     IRQ_service5
1150 0000F776 [B0410000]                  <1>                                dd     fdc_int
1151 0000F77A [BCF70000]                  <1>                                dd     IRQ_service7
1152 0000F77E [F6070000]                  <1>                                dd     rtc_int
1153 0000F782 [C6F70000]                  <1>                                dd     IRQ_service9
1154 0000F786 [D0F70000]                  <1>                                dd     IRQ_service10
1155 0000F78A [DAF70000]                  <1>                                dd     IRQ_service11
1156 0000F78E [E4F70000]                  <1>                                dd     IRQ_service12
1157 0000F792 [EEF70000]                  <1>                                dd     IRQ_service13
1158 0000F796 [2C4B0000]                  <1>                                dd     hdc1_int
1159 0000F79A [534B0000]                  <1>                                dd     hdc2_int
1160                                     <1>
1161                                     <1>                                ; 03/03/2017
1162                                     <1>                                ; 27/02/2017
1163                                     <1> IRQ_service3:
1164 0000F79E 36C605[444B0100]03          <1>                                mov    byte [ss:IRQnum], 3
1165 0000F7A6 EB4E                        <1>                                jmp     short IRQ_service
1166                                     <1> IRQ_service4:
1167 0000F7A8 36C605[444B0100]04          <1>                                mov    byte [ss:IRQnum], 4
1168 0000F7B0 EB44                        <1>                                jmp     short IRQ_service
1169                                     <1> IRQ_service5:
1170 0000F7B2 36C605[444B0100]05          <1>                                mov    byte [ss:IRQnum], 5
1171 0000F7BA EB3A                        <1>                                jmp     short IRQ_service
1172                                     <1> IRQ_service7:
1173 0000F7BC 36C605[444B0100]07          <1>                                mov    byte [ss:IRQnum], 7
1174 0000F7C4 EB30                        <1>                                jmp     short IRQ_service
1175                                     <1> IRQ_service9:
1176 0000F7C6 36C605[444B0100]09          <1>                                mov    byte [ss:IRQnum], 9
1177 0000F7CE EB26                        <1>                                jmp     short IRQ_service
1178                                     <1> IRQ_service10:
1179 0000F7D0 36C605[444B0100]0A          <1>                                mov    byte [ss:IRQnum], 10
1180 0000F7D8 EB1C                        <1>                                jmp     short IRQ_service
1181                                     <1> IRQ_service11:
1182 0000F7DA 36C605[444B0100]0B          <1>                                mov    byte [ss:IRQnum], 11
1183 0000F7E2 EB12                        <1>                                jmp     short IRQ_service
1184                                     <1> IRQ_service12:
1185 0000F7E4 36C605[444B0100]0C          <1>                                mov    byte [ss:IRQnum], 12
1186 0000F7EC EB08                        <1>                                jmp     short IRQ_service
1187                                     <1> IRQ_service13:
1188 0000F7EE 36C605[444B0100]0D          <1>                                mov    byte [ss:IRQnum], 13
1189                                     <1>                                ; jmp short IRQ_service
1190                                     <1> IRQ_service:
1191                                     <1>                                ; 04/03/2017
1192                                     <1>                                ; 01/03/2017
1193                                     <1>                                ; 28/02/2017
1194                                     <1>                                ; 27/02/2017
1195 0000F7F6 1E                          <1>                                push   ds
1196 0000F7F7 06                          <1>                                push   es
1197 0000F7F8 0FA0                        <1>                                push   fs
1198 0000F7FA 0FA8                        <1>                                push   gs
1199                                     <1>
1200 0000F7FC 60                          <1>                                pushad ; eax,ecx,edx,ebx,esp,ebp,esi,edi
1201 0000F7FD 66B91000                   <1>                                mov     cx, KDATA
1202 0000F801 8ED9                       <1>                                mov     ds, cx
1203 0000F803 8EC1                       <1>                                mov     es, cx
1204 0000F805 8EE1                       <1>                                mov     fs, cx
1205 0000F807 8EE9                       <1>                                mov     gs, cx
1206                                     <1>
1207 0000F809 0F20D8                     <1>                                mov     eax, cr3
1208 0000F80C A3[404B0100]               <1>                                mov     [IRQ_cr3], eax
1209                                     <1>
1210 0000F811 A1[20380100]               <1>                                mov     eax, [k_page_dir]
1211 0000F816 0F22D8                     <1>                                mov     cr3, eax
1212                                     <1>
1213 0000F819 A0[444B0100]               <1>                                mov     al, [IRQnum]
1214                                     <1>
1215                                     <1>                                ; mov cl, [sysflg]
1216                                     <1>                                ; mov [u.r_mode], cl ; system (0) or user mode (FFh)
1217                                     <1> IRQsrv_0:
1218 0000F81E 0FB6D8                     <1>                                movzx   ebx, al
1219 0000F821 8A9B[DC030100]             <1>                                mov     bl, [ebx+IRQenum] ; IRQ (available) index number + 1
1220                                     <1>                                ; 01/03/2017
1221 0000F827 FECB                       <1>                                dec     bl ; IRQ index number, 0 to 8
1222 0000F829 0F8807010000              <1>                                js      IRQsrv_5 ; not available to use here!?
1223                                     <1>                                ;
1224 0000F82F 80BB[0A4B0100]80           <1>                                cmp     byte [ebx+IRQ.method], 80h ; using by a dev or kernel?
1225 0000F836 7205                       <1>                                jnb     short IRQsrv_1 ; no
1226                                     <1>
1227                                     <1>                                ; If the IRQ service is already owned by TRDOS 386 kernel
1228                                     <1>                                ; or a Device driver
1229                                     <1>                                ; we need to call 'dev_IRQ_service'
1230                                     <1>
1231                                     <1>                                ; IRQ number in AL
1232 0000F838 E868020000                 <1>                                call    dev_IRQ_service ; IRQ service for device drivers
1233                                     <1>                                ; IRQ number in AL
1234                                     <1> IRQsrv_1:
1235                                     <1>                                ; check user callback service status
1236                                     <1>                                ; AL = IRQ number
1237                                     <1>                                ; EBX = IRQ (Available) Index number
1238                                     <1>

```

```

1239 0000F83D A2[1F4F0100] <1> mov [u.irqwait], al ; set waiting IRQ flag
1240 <1>
1241 0000F842 8A83[F84A0100] <1> mov al, [ebx+IRQ.owner]
1242 0000F848 20C0 <1> and al, al
1243 0000F84A 0F84E6000000 <1> jz IRQsrv_5 ; it is not owned by a user/proc
1244 <1>
1245 <1> ; 03/03/2017
1246 0000F850 89DA <1> mov edx, ebx
1247 0000F852 C0E202 <1> shl dl, 2
1248 0000F855 8B92[1C4B0100] <1> mov edx, [edx+IRQ.addr] ; S.R.B. or Callback service addr
1249 <1>
1250 0000F85B 8AA3[0A4B0100] <1> mov ah, [ebx+IRQ.method]
1251 0000F861 F6C401 <1> test ah, 1
1252 0000F864 7534 <1> jnz short IRQsrv_4 ; Callback service method
1253 <1>
1254 <1> ; Signal Response Byte method
1255 <1> ;mov edx, [edx+IRQ.addr] ; Signal Response Byte address
1256 <1> ; ; (Physical address, non-swappable)
1257 0000F866 80E402 <1> and ah, 2 ; bit 1, (S.R.B.) counter method
1258 0000F869 8AA3[134B0100] <1> mov ah, [ebx+IRQ.srb] ; Signal Response Byte value
1259 0000F86F 7408 <1> jz short IRQsrv_2 ; fixed S.R.B. value
1260 <1> ; counter method (increment)
1261 0000F871 FEC4 <1> inc ah
1262 0000F873 88A3[134B0100] <1> mov [ebx+IRQ.srb], ah ; next (count) number
1263 <1> IRQsrv_2:
1264 0000F879 8822 <1> mov [edx], ah ; put S.R.B. val to the user's S.R.B. addr
1265 0000F87B C605[1F4F0100]00 <1> mov byte [u.irqwait], 0 ; clear waiting IRQ flag
1266 <1>
1267 0000F882 3A05[FB4E0100] <1> cmp al, [u.uno]
1268 0000F888 0F84A8000000 <1> je IRQsrv_5 ; the owner is current user/process
1269 <1> IRQsrv_3:
1270 <1> ; the owner is not current user/process
1271 <1> ; AL = process number
1272 0000F88E B202 <1> mov dl, 2 ; priority, 2 = event (high)
1273 0000F890 E855FAFFFF <1> call set_run_sequence
1274 <1>
1275 <1> ; [u.irqwait] = waiting IRQ number for callback service
1276 <1>
1277 0000F895 E99C000000 <1> jmp IRQsrv_5
1278 <1> IRQsrv_4:
1279 0000F89A 3A05[FB4E0100] <1> cmp al, [u.uno]
1280 0000F8A0 75EC <1> jne short IRQsrv_3 ; the owner is current user/process
1281 <1>
1282 <1> ; Check if an IRQ callback service already in progress
1283 0000F8A2 803D[204F0100]00 <1> cmp byte [u.r_lock], 0
1284 0000F8A9 0F8787000000 <1> ja IRQsrv_5 ; nothing to do !
1285 <1> ; (we need to complete prev callback)
1286 <1>
1287 0000F8AF 803D[1C4F0100]00 <1> cmp byte [u.t_lock], 0
1288 0000F8B6 777E <1> ja short IRQsrv_5 ; nothing to do !
1289 <1> ; (we need to complete timer callback)
1290 <1>
1291 <1> ; 04/03/2017
1292 0000F8B8 C605[1F4F0100]00 <1> mov byte [u.irqwait], 0 ; reset/clear waiting IRQ flag
1293 <1>
1294 0000F8BF FE05[204F0100] <1> inc byte [u.r_lock] ; 'IRQ callback service in progress' flag
1295 <1>
1296 0000F8C5 8A0D[A14E0100] <1> mov cl, [sysflg] ; (system call) mode flag (kernel/user)
1297 0000F8CB 880D[214F0100] <1> mov [u.r_mode], cl ; system mode (0) or user mode (FFh)
1298 <1>
1299 <1> ;
1300 0000F8D1 8B2D[BC370100] <1> mov ebp, [tss.esp0] ; kernel stack address (for ring 0)
1301 0000F8D7 83ED14 <1> sub ebp, 20 ; eip, cs, eflags, esp, ss
1302 0000F8DA 892D[A44E0100] <1> mov [u.sp], ebp
1303 0000F8E0 8925[A84E0100] <1> mov [u.usp], esp
1304 <1>
1305 <1> ;or word [ebp+8], 200h ; 22/01/2017, force enabling interrupts
1306 <1>
1307 0000F8E6 8B44241C <1> mov eax, [esp+28] ; pushed eax
1308 0000F8EA A3[AC4E0100] <1> mov [u.r0], eax
1309 <1>
1310 0000F8EF E881EDFFFF <1> call wswap ; save user's registers & status
1311 <1>
1312 <1> ; software int is in ring 0 but IRQ handler must return to ring 3
1313 <1> ; so, ring 3 return address and stack registers
1314 <1> ; (eip, cs, eflags, esp, ss)
1315 <1> ; must be copied to IRQ handler return
1316 <1> ; eip will be replaced by callback service routine address
1317 <1>
1318 0000F8F4 C605[A14E0100]FF <1> mov byte [sysflg], 0FFh ; user mode
1319 <1>
1320 <1> ; system mode (system call)
1321 <1> ;mov ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),
1322 <1> ; ESP (u), SS (UDATA)
1323 <1>
1324 0000F8FB 8B4510 <1> mov eax, [ebp+16]; SS (UDATA)
1325 0000F8FE 89E6 <1> mov esi, esp
1326 0000F900 50 <1> push eax
1327 0000F901 50 <1> push eax
1328 0000F902 89E7 <1> mov edi, esp
1329 0000F904 893D[A84E0100] <1> mov [u.usp], edi
1330 0000F90A B908000000 <1> mov ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS
1331 0000F90F F3A5 <1> rep movsd
1332 0000F911 B104 <1> mov cl, 4
1333 0000F913 F3AB <1> rep stosd
1334 0000F915 893D[A44E0100] <1> mov [u.sp], edi
1335 0000F91B 89EE <1> mov esi, ebp
1336 0000F91D B105 <1> mov cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)
1337 0000F91F F3A5 <1> rep movsd
1338 <1> ;
1339 <1>
1340 0000F921 8B0D[004F0100] <1> mov ecx, [u.pgdir]

```

```

1341 0000F927 890D[404B0100] <1>          mov    [IRQ_cr3], ecx
1342 <1>
1343 <1> set_IRQ_callback_addr:
1344 <1>          ;
1345 <1>          ; This routine sets return address
1346 <1>          ; to start of user's interrupt
1347 <1>          ; service (callback) address
1348 <1>          ;
1349 <1>          ; INPUT:
1350 <1>          ;     EDX = callback routine/service address
1351 <1>          ;     (virtual, not physical address!)
1352 <1>          ;     [u.sp] = kernel stack, points to
1353 <1>          ;     user's EIP,CS,EFLAGS,ESP,SS
1354 <1>          ;     registers.
1355 <1>          ; OUTPUT:
1356 <1>          ;     EIP (user) = callback (service) address
1357 <1>          ;     CS (user) = UCODE
1358 <1>          ;     EFLAGS (user) = flags before callback
1359 <1>          ;     ESP (user) = ESP-4 (user, before callback)
1360 <1>          ;     [ESP](user) = EIP (user) before callback
1361 <1>          ;
1362 <1>          ; Note: If CPU was in user mode while entering
1363 <1>          ;     the timer interrupt service routine,
1364 <1>          ;     'IRET' will get return to callback routine
1365 <1>          ;     immediately. If CPU was in system/kernel mode
1366 <1>          ;     'iret' will get return to system call and
1367 <1>          ;     then, callback routine will be return address
1368 <1>          ;     from system call. (User's callback/service code
1369 <1>          ;     will be able to return to normal return address
1370 <1>          ;     via a 'sysrele' system call at the end.)
1371 <1>          ;
1372 <1>          ; Note: User's IRQ callback service code must be ended
1373 <1>          ;     with a 'sysrele' system call !
1374 <1>          ;
1375 <1>          ; For example:
1376 <1>          ;
1377 <1>          ;     audio_IRQ_callback:
1378 <1>          ;     ...
1379 <1>          ;     <load DMA buffer with audio data>
1380 <1>          ;     ...
1381 <1>          ;     mov eax, 39 ; 'sysrele'
1382 <1>          ;     int 40h ; TRDOS 386 system call (interrupt)
1383 <1>          ;
1384 <1>
1385 <1>          ;mov  edx, [edx+IRQ.addr] ; Callback service address
1386 <1>          ;          ; (Virtual address)
1387 <1>
1388 0000F92D 8B2D[A44E0100] <1>          mov    ebp, [u.sp]; kernel's stack, points to EIP (user)
1389 0000F933 895500 <1>          mov    [ebp], edx
1390 <1> IRQsrv_5:
1391 <1>          ; EOI & return
1392 0000F936 803D[444B0100]07 <1>          cmp    byte [IRQnum], 7
1393 0000F93D 7604 <1>          jna    short IRQsrv_6
1394 <1>          ;
1395 0000F93F B020 <1>          mov    al, EOI          ; end of interrupt
1396 <1>          ;cli    ; disable interrupts till stack cleared
1397 0000F941 E6A0 <1>          out    INTB00, al ; For controll2 #2
1398 <1> IRQsrv_6:
1399 <1>          ;mov  byte [IRQnum], 0 ; reset
1400 0000F943 B020 <1>          mov    al, EOI          ; end of interrupt
1401 0000F945 FA <1>          cli    ; disable interrupts till stack cleared
1402 0000F946 E620 <1>          out    INTA00, al ; end of interrupt to 8259 - 1
1403 <1> IRQsrv_7:
1404 0000F948 8B0D[404B0100] <1>          mov    ecx, [IRQ_cr3]      ; previous content of cr3 register
1405 0000F94E 0F22D9 <1>          mov    cr3, ecx          ; restore cr3 register content
1406 <1>          ;
1407 0000F951 61 <1>          popad ; edi,esi,ebp,(increment esp by 4), ebx,edx,ecx,eax
1408 <1>          ;
1409 0000F952 0FA9 <1>          pop    gs
1410 0000F954 0FA1 <1>          pop    fs
1411 0000F956 07 <1>          pop    es
1412 0000F957 1F <1>          pop    ds
1413 <1>          ;
1414 0000F958 CF <1>          iretd ; return from interrupt
1415 <1>
1416 <1> get_device_number:
1417 <1>          ; 08/10/2016
1418 <1>          ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1419 <1>          ;
1420 <1>          ; This procedure compares name of requested
1421 <1>          ; device with kernel device names and
1422 <1>          ; installable device names. If names match,
1423 <1>          ; the relevant device index (entry) number
1424 <1>          ; will be returned the caller (sysopen)
1425 <1>          ; for the requested device.
1426 <1>          ;
1427 <1>          ; NOTE: Installable device drivers must
1428 <1>          ; be loaded before using 'sysopen'
1429 <1>          ; (opendev) system call.
1430 <1>          ;
1431 <1>          ; INPUT:
1432 <1>          ;     ESI = device name address (ASCIIIZ)
1433 <1>          ;     (in kernel's memory space)
1434 <1>          ;     max name length = 8 without '/dev/'
1435 <1>          ;     Device name will be capitalized
1436 <1>          ;     and if there is, '/dev/' will be
1437 <1>          ;     removed from name before comparising)
1438 <1>          ;
1439 <1>          ; OUTPUT:
1440 <1>          ;     cf = 0 ->
1441 <1>          ;     EAX (AL) = device entry/index number
1442 <1>          ;     cf = 1 -> device not found (installed)

```



```
1443 <1> ; or invalid device name
1444 <1> ; (AL=0)
1445 <1> ; device_name = device name address (asciiz)
1446 <1> ;
1447 <1> ; Modified registers: EAX, EBX, ESI, EDI
1448 <1>
1449 0000F959 BF[E1450100] <1> mov edi, device_name
1450 0000F95E E805010000 <1> call lodsb_capitalize
1451 0000F963 88C4 <1> mov ah, al
1452 0000F965 3C2F <1> cmp al, '/'
1453 0000F967 750E <1> jne short gdn_1
1454 0000F969 BF[E1450100] <1> mov edi, device_name
1455 0000F96E E8F5000000 <1> call lodsb_capitalize
1456 <1> gdn_0:
1457 0000F973 20C0 <1> and al, al ; 0 ?
1458 0000F975 7420 <1> jz short gdn_err ; null name after '/'
1459 <1> gdn_1:
1460 0000F977 3C44 <1> cmp al, 'D'
1461 0000F979 7517 <1> jne short gdn_2
1462 0000F97B E8E8000000 <1> call lodsb_capitalize
1463 0000F980 3C45 <1> cmp al, 'E'
1464 0000F982 750E <1> jne short gdn_2
1465 0000F984 E8DF000000 <1> call lodsb_capitalize
1466 0000F989 3C56 <1> cmp al, 'V'
1467 0000F98B 7505 <1> jne short gdn_2
1468 0000F98D AC <1> lodsb
1469 0000F98E 3C2F <1> cmp al, '/'
1470 0000F990 740D <1> je short gdn_4
1471 <1> gdn_2:
1472 0000F992 80FC2F <1> cmp ah, '/'
1473 0000F995 750F <1> jne short gdn_5
1474 <1> gdn_err:
1475 <1> ; invalid device name or device not found
1476 0000F997 31C0 <1> xor eax, eax ; 0
1477 0000F999 F9 <1> stc
1478 0000F99A C3 <1> retn
1479 <1> gdn_3:
1480 0000F99B 3C2F <1> cmp al, '/'
1481 0000F99D 7507 <1> jne short gdn_5
1482 <1> gdn_4:
1483 0000F99F BF[E1450100] <1> mov edi, device_name
1484 0000F9A4 EB04 <1> jmp short gdn_6
1485 <1> gdn_5:
1486 0000F9A6 3C00 <1> cmp al, 0
1487 0000F9A8 7419 <1> je short gdn_7
1488 <1> gdn_6:
1489 0000F9AA E8B9000000 <1> call lodsb_capitalize
1490 0000F9AF 81FF[E9450100] <1> cmp edi, device_name + 8
1491 0000F9B5 72E4 <1> jb short gdn_3
1492 0000F9B7 3C00 <1> cmp al, 0
1493 0000F9B9 75DC <1> jne short gdn_err
1494 0000F9BB 81FF[E2450100] <1> cmp edi, device_name + 1
1495 0000F9C1 76D4 <1> jna short gdn_err ; null name after '/'
1496 <1> gdn_7:
1497 0000F9C3 AA <1> stosb
1498 <1> ; zero padding ("NAME",0,0,0,0)
1499 0000F9C4 81FF[E9450100] <1> cmp edi, device_name + 8
1500 0000F9CA 72F7 <1> jb short gdn_7
1501 <1> gdn_8:
1502 <1> ; search for kernel device names
1503 0000F9CC BE[E1450100] <1> mov esi, device_name
1504 0000F9D1 BF[C2010100] <1> mov edi, KDEV_NAME
1505 0000F9D6 31C0 <1> xor eax, eax
1506 <1> gdn_9:
1507 0000F9D8 A7 <1> cmpsd
1508 0000F9D9 7505 <1> jne short gdn_10
1509 0000F9DB A7 <1> cmpsd
1510 0000F9DC 7503 <1> jne short gdn_11
1511 0000F9DE EB2B <1> jmp short gdn_17 ; match
1512 <1> gdn_10:
1513 0000F9E0 A7 <1> cmpsd ; add esi, 4 & add edi, 4
1514 <1> gdn_11:
1515 0000F9E1 BE[E1450100] <1> mov esi, device_name
1516 0000F9E6 FEC0 <1> inc al
1517 0000F9E8 3C16 <1> cmp al, NumOfKernelDevNames
1518 0000F9EA 72EC <1> jb short gdn_9
1519 <1> gdn_12:
1520 <1> ; search for installable device names
1521 <1> ; esi = offset device_name
1522 0000F9EC BF[0C460100] <1> mov edi, IDEV_NAME
1523 0000F9F1 28C0 <1> sub al, al ; 0
1524 <1> gdn_13:
1525 0000F9F3 A7 <1> cmpsd
1526 0000F9F4 7505 <1> jne short gdn_14
1527 0000F9F6 A7 <1> cmpsd
1528 0000F9F7 7503 <1> jne short gdn_15
1529 0000F9F9 EB3F <1> jmp short gdn_19 ; match
1530 <1> gdn_14:
1531 0000F9FB A7 <1> cmpsd ; add esi, 4 & add edi, 4
1532 <1> gdn_15:
1533 0000F9FC BE[E1450100] <1> mov esi, device_name
1534 0000FA01 FEC0 <1> inc al
1535 0000FA03 3C08 <1> cmp al, NumOfInstallableDevices
1536 0000FA05 72EC <1> jb short gdn_13
1537 <1>
1538 <1> gdn_16:
1539 0000FA07 30C0 <1> ; error: invalid device name (not found) !
1540 0000FA09 F9 <1> xor al, al
1541 0000FA0A C3 <1> stc
1542 <1> retn
1543 <1> gdn_17:
1544 <1> ; ; name match (with one of kernel device names)
; ;
```

```

1545 <1> ; convert KDEV_NAME index to
1546 <1> ; KDEV_NUMBER index
1547 <1> ; (different names are used for same devices)
1548 <1> ; (example: "COM1" & "TTY8" = device number 18)
1549 0000FA0B 89C3 <1> mov ebx, eax ; < 256
1550 0000FA0D 8A83[72020100] <1> mov al, [KDEV_NUMBER+ebx]
1551 <1>
1552 <1> ; check if empty dev entry in the list
1553 0000FA13 80B8[90470100]00 <1> cmp byte [DEV_OPENMODE+eax], 0
1554 0000FA1A 771B <1> ja short gdn_18 ; it must be already set
1555 <1>
1556 <1> ; (re)set device name and access flags
1557 <1> ; (remain open work will be easy after that)
1558 <1> ; (NOTE: here, data will be copied to bss section)
1559 0000FA1C 88C3 <1> mov bl, al
1560 0000FA1E 83EF08 <1> sub edi, 8 ; kernel device name address (data)
1561 0000FA21 66C1E302 <1> shl bx, 2
1562 0000FA25 89BB[AE470100] <1> mov [DEV_NAME_PTR+ebx], edi ; (all) device names
1563 0000FA2B 8A98[C8030100] <1> mov bl, [KDEV_ACCESS+eax] ; kernel dev list (data)
1564 0000FA31 8898[DC460100] <1> mov [DEV_ACCESS+eax], bl ; (all) device list (bss)
1565 <1> gdn_18:
1566 0000FA37 FEC0 <1> inc al ; 1 to NumOfKernelDevNames (<=7Fh)
1567 <1> ; eax = device index/entry number
1568 0000FA39 C3 <1> retn
1569 <1>
1570 <1> gdn_19: ; name match (with one of installable device names)
1571 <1> ;
1572 <1> ; al = 0 to NumOfInstallableDevices - 1 (<=7Fh)
1573 <1>
1574 0000FA3A 89C3 <1> mov ebx, eax
1575 0000FA3C 80C316 <1> add bl, NumOfKernelDevices ; < NUMOFDEVICES
1576 <1>
1577 <1> ; check if empty dev entry in the list
1578 0000FA3F 80BB[90470100]00 <1> cmp byte [DEV_OPENMODE+ebx], 0
1579 0000FA46 771D <1> ja short gdn_20 ; it must be already set
1580 <1>
1581 <1> ; (re)set device name and access flags
1582 <1> ; (remain open work will be easy after that)
1583 0000FA48 83EF08 <1> sub edi, 8 ; installable device name address
1584 0000FA4B 66C1E302 <1> shl bx, 2 ; *4
1585 0000FA4F 89BB[AE470100] <1> mov [DEV_NAME_PTR+ebx], edi ; (all) device names
1586 0000FA55 66C1EB02 <1> shr bx, 2
1587 0000FA59 8A80[54460100] <1> mov al, [IDEV_FLAGS+eax] ; installable dev list
1588 0000FA5F 8883[DC460100] <1> mov [DEV_ACCESS+ebx], al ; (all) device list
1589 <1> gdn_20:
1590 0000FA65 88D8 <1> mov al, bl
1591 <1> ; eax = device index/entry number ; < NUMOFDEVICES
1592 0000FA67 C3 <1> retn
1593 <1>
1594 <1> lods_sb_capitalize:
1595 <1> ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1596 <1> ; INPUT -> [esi] = character
1597 <1> ; edi = destination
1598 <1> ; OUTPUT -> AL contains capitalized character
1599 <1> ; esi = esi+1
1600 <1> ; edi = edi+1
1601 <1> ;
1602 0000FA68 AC <1> lods_b
1603 0000FA69 3C61 <1> cmp al, 61h
1604 0000FA6B 7206 <1> jnb short lods_sb_cap_retn
1605 0000FA6D 3C7A <1> cmp al, 7Ah
1606 0000FA6F 7702 <1> ja short lods_sb_cap_retn
1607 0000FA71 24DF <1> and al, 0DFh
1608 <1> lods_sb_cap_retn:
1609 0000FA73 AA <1> stos_b
1610 0000FA74 C3 <1> retn
1611 <1>
1612 <1> device_open:
1613 <1> ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)
1614 <1> ; Complete device opening work for sysopen (device)
1615 <1> ;
1616 <1> ; INPUT ->
1617 <1> ; EAX = Device Number (AL)
1618 <1> ; CL = Open mode (1 = read, 2 = write)
1619 <1> ; CH = Device access byte (bit 0 = 0)
1620 <1> ; OUTPUT ->
1621 <1> ; EAX = Device Number
1622 <1> ; CF = 0 -> device has been opened
1623 <1> ; CF = 1 -> device could not be opened
1624 <1> ;
1625 <1> ; Modified registers: ebx, (edx, ecx, esi, edi, ebp)
1626 <1> ;
1627 <1>
1628 0000FA75 89C3 <1> mov ebx, eax
1629 0000FA77 66C1E302 <1> shl bx, 2 ; *4
1630 <1>
1631 0000FA7B F6C580 <1> test ch, 80h ; bit 7, installable device driver flag
1632 0000FA7E 7406 <1> jz short d_open_2 ; Kernel device
1633 <1> ; installable device
1634 <1> d_open_1:
1635 0000FA80 FFA3[58460100] <1> jmp dword [ebx+IDEV_OADDR-4]
1636 <1> d_open_2:
1637 0000FA86 FFA3[84020100] <1> jmp dword [ebx+KDEV_OADDR-4]
1638 <1>
1639 <1> device_close:
1640 <1> ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)
1641 <1> ; Complete device closing work for sysclose (device)
1642 <1> ;
1643 <1> ; INPUT ->
1644 <1> ; EAX = Device Number (AL)
1645 <1> ; CL = Open mode (1 = read, 2 = write)
1646 <1> ; CH = Device access byte (bit 0 = 0)

```

```

1647 <1> ; OUTPUT ->
1648 <1> ; EAX = Device Number
1649 <1> ; CF = 0 -> device has been closed
1650 <1> ; CF = 1 -> device could not be closed
1651 <1> ;
1652 <1> ; Modified registers: ebx, (edx, ecx, esi, edi, ebp)
1653 <1> ;
1654 <1>
1655 0000FA8C 89C3 <1> mov ebx, eax
1656 0000FA8E 66C1E302 <1> shl bx, 2 ; *4
1657 <1>
1658 0000FA92 F6C580 <1> test ch, 80h ; bit 7, installable device driver flag
1659 0000FA95 7406 <1> jz short d_close_2 ; Kernel device
1660 <1> ; installable device
1661 <1> d_close_1:
1662 0000FA97 FFA3[78460100] <1> jmp dword [ebx+IDEV_CADDR-4]
1663 <1> d_close_2:
1664 0000FA9D FFA3[D4020100] <1> jmp dword [ebx+KDEV_CADDR-4]
1665 <1>
1666 <1> rnull:
1667 <1> ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1668 <1> ; read null (read from null device)
1669 0000FAA3 C3 <1> retn
1670 <1>
1671 <1> wnull:
1672 <1> ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1673 <1> ; write null (write to null device)
1674 0000FAA4 C3 <1> retn
1675 <1>
1676 <1> dev_IRQ_service:
1677 <1> ; 27/02/2017 - TRDOS 386 (TRDOS v2.0)
1678 <1> ; INPUT ->
1679 <1> ; AL = IRQ Number
1680 <1> ;
1681 <1> ;push ebx
1682 <1> ;push eax
1683 <1> ; .....
1684 <1> ;pop eax
1685 <1> ;pop ebx
1686 0000FAA5 C3 <1> retn ; nothing to do for now ; 27/02/2017
1687 <1>
1688 <1> otty:
1689 <1> sret:
1690 <1> ocvt:
1691 <1> ctty:
1692 <1> cret:
1693 <1> ccvt:
1694 <1> rtty:
1695 <1> wtty:
1696 <1> rmem:
1697 <1> wmem:
1698 <1> rfd:
1699 <1> rhd:
1700 <1> wfd:
1701 <1> whd:
1702 <1> rlpt:
1703 <1> wlpt:
1704 <1> rcvt:
1705 <1> xmtt:
1706 0000FAA6 C3 <1> retn
2312 <1> %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: 26/03/2017
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 0000FAA7 01 <1> db 1 ; A: = 0 & B: = 1
22 <1>
23 <1> Restore_CDIR:
24 0000FAA8 FF <1> db 0FFh ; Initial value -> any number except 0
25 <1>
26 <1> msg_CRLF_temp:
27 0000FAA9 070D0A00 <1> db 07h, 0Dh, 0Ah, 0
28 <1>
29 <1> Magic_Bytes:
30 0000FAAD 04 <1> db 4
31 0000FAAE 01 <1> db 1
32 <1> mainprog_Version:
33 0000FAAF 07 <1> db 7
34 0000FAB0 5B5452444F535D204D- <1> db "[TRDOS] Main Program v2.0.260317"
34 0000FAB9 61696E2050726F6772- <1>
34 0000FAC2 616D2076322E302E32- <1>
34 0000FACB 3630333137 <1>
35 0000FAD0 0D0A <1> db 0Dh, 0Ah
36 0000FAD2 286329204572646F67- <1> db "(c) Erdogan Tan 2005-2017"
36 0000FADB 616E2054616E203230- <1>
36 0000FAE4 30352D32303137 <1>

```

```
37 0000FAEB 0D0A00 <1> db 0Dh, 0Ah, 0
38 <1>
39 <1> MainProgCfgFile: ; 14/04/2016
40 0000FAEE 4D41494E50524F472E- <1> db "MAINPROG.CFG", 0
40 0000FAF7 43464700 <1>
41 <1>
42 <1> TRDOSPromptLabel:
43 0000FAFB 5452444F53 <1> db "TRDOS"
44 0000FB00 00 <1> db 0
45 0000FB01 00<rept> <1> times 5 db 0
46 0000FB06 00 <1> db 0
47 <1>
48 <1> ; INTERNAL COMMANDS
49 <1> Command_List:
50 0000FB07 44495200 <1> Cmd_Dir: db "DIR", 0
51 0000FB0B 434400 <1> Cmd_Cd: db "CD", 0
52 0000FB0E 433A00 <1> Cmd_Drive: db "C:", 0
53 0000FB11 56455200 <1> Cmd_Ver: db "VER", 0
54 0000FB15 4558495400 <1> Cmd_Exit: db "EXIT", 0
55 0000FB1A 50524F4D505400 <1> Cmd_Prompt: db "PROMPT", 0
56 0000FB21 564F4C554D4500 <1> Cmd_Volume: db "VOLUME", 0
57 0000FB28 4C4F4E474E414D4500 <1> Cmd_LongName: db "LONGNAME", 0
58 0000FB31 4441544500 <1> Cmd_Date: db "DATE", 0
59 0000FB36 54494D4500 <1> Cmd_Time: db "TIME", 0
60 0000FB3B 52554E00 <1> Cmd_Run: db "RUN", 0
61 0000FB3F 53455400 <1> Cmd_Set: db "SET", 0
62 0000FB43 434C5300 <1> Cmd_Cls: db "CLS", 0
63 0000FB47 53484F5700 <1> Cmd_Show: db "SHOW", 0
64 0000FB4C 44454C00 <1> Cmd_Del: db "DEL", 0
65 0000FB50 41545452494200 <1> Cmd_Attrib: db "ATTRIB", 0
66 0000FB57 52454E414D4500 <1> Cmd_Rename: db "RENAME", 0
67 0000FB5E 524D44495200 <1> Cmd_Rmdir: db "RMDIR", 0
68 0000FB64 4D4B44495200 <1> Cmd_Mkdir: db "MKDIR", 0
69 0000FB6A 434F505900 <1> Cmd_Copy: db "COPY", 0
70 0000FB6F 4D4F564500 <1> Cmd_Move: db "MOVE", 0
71 0000FB74 5041544800 <1> Cmd_Path: db "PATH", 0
72 0000FB79 4D454D00 <1> Cmd_Mem: db "MEM", 0
73 0000FB7D 00 <1> db 0
74 0000FB7E 46494E4400 <1> Cmd_Find: db "FIND", 0
75 0000FB83 4543484F00 <1> Cmd_Echo: db "ECHO", 0
76 0000FB88 2A00 <1> Cmd_Remark: db "!", 0
77 0000FB8A 3F00 <1> Cmd_Help: db "?", 0
78 0000FB8C 44455649434500 <1> Cmd_Device: db "DEVICE", 0
79 0000FB93 4445564C49535400 <1> Cmd_DevList: db "DEVLIST", 0
80 0000FB9B 434844495200 <1> Cmd_Chdir: db "CHDIR", 0
81 0000FBA1 4245455000 <1> Cmd_Beep: db "BEEP", 0
82 <1>
83 0000FBA6 00 <1> db 0
84 <1>
85 <1> ; 15/02/2016 (FILE.ASM, 09/10/2011)
86 <1> invalid_fname_chars:
87 0000FBA7 222728292A2B2C2F <1> db 22h, 27h, 28h, 29h, 2Ah, 2Bh, 2Ch, 2Fh
88 0000FBAF 3A3B3C3D3E3F40 <1> db 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh, 40h
89 0000FBB6 5B5C5D5E60 <1> db 5Bh, 5Ch, 5Dh, 5Eh, 60h
90 <1> sizeInvFnChars equ ($ - invalid_fname_chars)
91 <1> ;
92 <1>
93 <1> Msg_Enter_Date:
94 0000FBBB 456E746572206E6577- <1> db 'Enter new date (dd-mm-yy): '
94 0000FBC4 206461746520286464- <1>
94 0000FBCD 2D6D6D2D7979293A20 <1>
95 0000FBD6 00 <1> db 0
96 <1> Msg_Show_Date:
97 0000FBD7 43757272656E742064- <1> db 'Current date is '
97 0000FBE0 61746520697320 <1>
98 0000FBE7 30 <1> Day: db '0'
99 0000FBE8 30 <1> db '0'
100 0000FBE9 2F <1> db '/'
101 0000FBEA 30 <1> Month: db '0'
102 0000FBEB 30 <1> db '0'
103 0000FBEC 2F <1> db '/'
104 0000FBED 30 <1> Century: db '0'
105 0000FBEE 30 <1> db '0'
106 0000FBEF 30 <1> Year: db '0'
107 0000FBF0 30 <1> db '0'
108 0000FBF1 0D0A00 <1> db 0Dh, 0Ah, 0
109 <1>
110 <1> Msg_Enter_Time:
111 0000FBF4 456E746572206E6577- <1> db 'Enter new time: '
111 0000FBFD 2074696D653A20 <1>
112 0000FC04 00 <1> db 0
113 <1> Msg_Show_Time:
114 0000FC05 43757272656E742074- <1> db 'Current time is '
114 0000FC0E 696D6520697320 <1>
115 0000FC15 30 <1> Hour: db '0'
116 0000FC16 30 <1> db '0'
117 0000FC17 3A <1> db ':'
118 0000FC18 30 <1> Minute: db '0'
119 0000FC19 30 <1> db '0'
120 0000FC1A 3A <1> db ':'
121 0000FC1B 30 <1> Second: db '0'
122 0000FC1C 30 <1> db '0'
123 0000FC1D 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 0000FC20 206B696C6F62797465- <1> db " kilobytes", 0Dh, 0Ah, 0
129 0000FC29 730D0A00 <1>
130 <1> VolSize_Bytes:
131 0000FC2D 2062797465730D0A00 <1> db " bytes", 0Dh, 0Ah, 0
```



```
132                                     <1> Volume_in_drive:
133 0000FC36 0D0A                       <1>             db 0Dh, 0Ah
134                                     <1> Vol_FS_Name:
135 0000FC38 54522046533120             <1>             db "TR FS1 "
136 0000FC3F 566F6C756D6520696E-       <1>             db "Volume in drive "
136 0000FC48 20647269766520             <1>
137 0000FC4F 30                         <1> Vol_Drv_Name:   db 30h
138 0000FC50 3A                         <1>             db ":"
139 0000FC51 20697320                   <1>             db " is "
140 0000FC55 0D0A00                     <1>             db 0Dh, 0Ah, 0
141                                     <1> Dir_Drive_Str:
142 0000FC58 54522D444F53204472-       <1>             db "TR-DOS Drive "
142 0000FC61 69766520                   <1>
143                                     <1> Dir_Drive_Name:
144 0000FC65 303A                       <1>             db "0:"
145 0000FC67 0D0A                       <1>             db 0Dh, 0Ah
146                                     <1> Vol_Str_Header:
147 0000FC69 566F6C756D65204E61-       <1>             db "Volume Name: "
147 0000FC72 6D653A20                   <1>
148                                     <1> Vol_Name:
149 0000FC76 00<rept>                   <1>             times 64 db 0
150 0000FCB6 00                         <1>             db 0
151                                     <1> Vol_Serial_Header:
152 0000FCB7 0D0A                       <1>             db 0Dh, 0Ah
153 0000FCB9 566F6C756D65205365-       <1>             db "Volume Serial No: "
153 0000FCC2 7269616C204E6F3A20       <1>
154                                     <1> Vol_Serial1:
155 0000FCCB 30303030                   <1>             db "0000"
156 0000FCCF 2D                         <1>             db "-"
157                                     <1> Vol_Serial2:
158 0000FCD0 30303030                   <1>             db "0000"
159 0000FCD4 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 0000FCD7 0D0A                       <1>             db 0Dh, 0Ah
165 0000FCD9 566F6C756D65205369-       <1>             db "Volume Size : ", 0
165 0000FCE2 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 0000FCE8 467265652053706163-       <1>             db "Free Space : ", 0
173 0000FCF1 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 0000FCF7 4469726563746F7279-       <1>             db "Directory: "
180 0000FD00 3A20                       <1>
181 0000FD02 2F                         <1> Dir_Str_Root:   db "/"
182 0000FD03 00<rept>                   <1> Dir_Str:       times 64 db 0
183 0000FD43 00000000                   <1>             dd 0
184 0000FD47 00                         <1>             db 0
185                                     <1>
186                                     <1> Msg_Bad_Command:
187 0000FD48 42616420636F6D6D61-       <1>             db "Bad command or file name!"
187 0000FD51 6E64206F7220666696C-     <1>
187 0000FD5A 65206E616D6521             <1>
188 0000FD61 0D0A00                     <1>             db 0Dh, 0Ah, 0
189                                     <1>
190                                     <1> msgl_drv_not_ready:
191 0000FD64 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 0000FD67 4472697665206E6F74-       <1>             db "Drive not ready or read error!"
196 0000FD70 207265616479206F72-       <1>
196 0000FD79 207265616420657272-       <1>
196 0000FD82 6F7221                     <1>
197 0000FD85 0D0A00                     <1>             db 0Dh, 0Ah, 0
198                                     <1>
199                                     <1> Msg_Not_Ready_Write_Err:
200 0000FD88 4472697665206E6F74-       <1>             db "Drive not ready or write error!"
200 0000FD91 207265616479206F72-       <1>
200 0000FD9A 207772697465206572-       <1>
200 0000FDA3 726F7221                   <1>
201 0000FDA7 0D0A00                     <1>             db 0Dh, 0Ah, 0
202                                     <1>
203                                     <1> Msg_Dir_Not_Found:
204 0000FDAA 4469726563746F7279-       <1>             db "Directory not found!"
204 0000FDB3 206E6F7420666F756E-       <1>
204 0000FDBC 6421                       <1>
205 0000FDBE 0D0A00                     <1>             db 0Dh, 0Ah, 0
206                                     <1>
207                                     <1> Msg_File_Not_Found:
208 0000FDC1 46696C65206E6F7420-       <1>             db "File not found!"
208 0000FDCA 666F756E6421               <1>
209 0000FDD0 0D0A00                     <1>             db 0Dh, 0Ah, 0
210                                     <1>
211                                     <1> Msg_File_Directory_Not_Found:
212 0000FDD3 46696C65206F722064-       <1>             db "File or directory not found!"
212 0000FDDC 69726563746F727920-       <1>
212 0000FDE5 6E6F7420666F756E64-       <1>
212 0000FDEE 21                         <1>
```

```

213 0000FDEF 0D0A00      <1>                db 0Dh, 0Ah, 0
214                                <1>
215                                <1> Msg_LongName_Not_Found:
216 0000FDF2 4C6F6E67206E616D65- <1>                db "Long name not found!"
216 0000FDFB 206E6F7420666F756E- <1>
216 0000FE04 6421        <1>
217 0000FE06 0D0A00      <1>                db 0Dh, 0Ah, 0
218                                <1>
219                                <1> beep_Insufficient_Memory: ; 20/02/2017
220 0000FE09 0D0A        <1>                db 0Dh, 0Ah
221 0000FE0B 07          <1>                db 07h
222                                <1> Msg_Insufficient_Memory:
223 0000FE0C 496E737566666696369- <1>                db "Insufficient memory!"
223 0000FE15 656E74206D656D6F72- <1>
223 0000FE1E 7921        <1>
224 0000FE20 0D0A00      <1>                db 0Dh, 0Ah, 0
225                                <1>
226                                <1> Msg_Error_Code:
227 0000FE23 436F6D6D616E642066- <1>                db 'Command failed! Error code : '
227 0000FE2C 61696C656421204572- <1>
227 0000FE35 726F7220636F646520- <1>
227 0000FE3E 3A20        <1>
228 0000FE40 303068      <1> error_code_hex: db '00h'
229 0000FE43 0A0A00      <1>                db 0Ah, 0Ah, 0
230                                <1>
231                                <1> align 2
232                                <1>
233                                <1> ; 10/02/2016
234                                <1> ; DIR.ASM - 09/10/2011
235                                <1>
236 0000FE46 3C4449523E20202020- <1> Type_Dir:      db '<DIR>' ; 10 bytes
236 0000FE4F 20          <1>
237                                <1>
238                                <1> File_Name:
239 0000FE50 20<rept>      <1>                times 12 db 20h
240 0000FE5C 20          <1>                db 20h
241                                <1> Dir_Or_FileSize:
242 0000FE5D 20<rept>      <1>                times 10 db 20h
243 0000FE67 20          <1>                db 20h
244                                <1> File_Attribute:
245 0000FE68 20202020      <1>                dd 20202020h
246 0000FE6C 20          <1>                db 20h
247                                <1> File_Day:
248 0000FE6D 3030        <1>                db '0','0'
249 0000FE6F 2F          <1>                db '/'
250                                <1> File_Month:
251 0000FE70 3030        <1>                db '0','0'
252 0000FE72 2F          <1>                db '/'
253                                <1> File_Year:
254 0000FE73 30303030      <1>                db '0','0','0','0'
255 0000FE77 20          <1>                db 20h
256                                <1> File_Hour:
257 0000FE78 3030        <1>                db '0','0'
258 0000FE7A 3A          <1>                db ':'
259                                <1> File_Minute:
260 0000FE7B 3030        <1>                db '0','0'
261 0000FE7D 00          <1>                db 0
262                                <1>
263                                <1> Decimal_File_Count_Header:
264 0000FE7E 0D0A        <1>                db 0Dh, 0Ah
265                                <1> Decimal_File_Count:
266 0000FE80 00<rept>      <1>                times 6 db 0
267                                <1>
268 0000FE86 2066696C6528732920- <1> str_files:      db " file(s) & "
268 0000FE8F 2620        <1>
269                                <1> Decimal_Dir_Count:
270 0000FE91 00<rept>      <1>                times 6 db 0
271                                <1> str_dirs:
272 0000FE97 206469726563746F72- <1>                db " directory(s) "
272 0000FEA0 7928732920      <1>
273 0000FEA5 0D0A00      <1>                db 0Dh, 0Ah, 0
274                                <1>
275 0000FEA8 206279746528732920- <1> str_bytes:      db " byte(s) in file(s)"
275 0000FEB1 696E2066696C652873- <1>
275 0000FEB4 29          <1>
276 0000FEBB 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 0000FEBE 496E76616C69642066- <1>                db "Invalid file or directory name characters!"
281 0000FEC7 696C65206F72206469- <1>
281 0000FED0 726563746F7279206E- <1>
281 0000FED9 616D65206368617261- <1>
281 0000FEE2 637465727321      <1>
282 0000FEE8 0D0A00      <1>                db 0Dh, 0Ah, 0
283                                <1> ; 21/02/2016
284 0000FEEB 46696C65206F722064- <1> Msg_Name_Exists: db "File or directory name exists!"
284 0000FEF4 69726563746F727920- <1>
284 0000FEFD 6E616D652065786973- <1>
284 0000FF06 747321      <1>
285 0000FF09 0D0A00      <1>                db 0Dh, 0Ah, 0
286                                <1> Msg_DoYouWantMkdir:
287 0000FF0C 446F20796F75207761- <1>                db "Do you want to make directory ", 0
287 0000FF15 6E7420746F206D616B- <1>
287 0000FF1E 65206469726563746F- <1>
287 0000FF27 72792000      <1>
288 0000FF2B 2028592F4E29203F20- <1> Msg_YesNo:      db " (Y/N) ? ", 0
288 0000FF34 00          <1>
289 0000FF35 000D0A00      <1> Y_N_nextline:    db 0, 0Dh, 0Ah, 0
290 0000FF39 4F4B2E0D0A00 <1> Msg_OK:          db "OK.", 0Dh, 0Ah, 0
291                                <1>

```

```
292                                     <1> ; 27/02/2016
293                                     <1> Msg_DoYouWantRmDir:
294 0000FF3F 446F20796F75207761- <1> db "Do you want to delete directory ", 0
294 0000FF48 6E7420746F2064656C- <1>
294 0000FF51 657465206469726563- <1>
294 0000FF5A 746F72792000 <1>
295                                     <1> Msg_Dir_Not_Empty:
296 0000FF60 4469726563746F7279- <1> db "Directory not empty!"
296 0000FF69 206E6F7420656D7074- <1>
296 0000FF72 7921 <1>
297 0000FF74 0D0A00 <1> db 0Dh, 0Ah, 0
298                                     <1>
299                                     <1> Msg_DoYouWantDelete:
300 0000FF77 446F20796F75207761- <1> db "Do you want to delete file ",0
300 0000FF80 6E7420746F2064656C- <1>
300 0000FF89 6574652066696C6520- <1>
300 0000FF92 00 <1>
301                                     <1>
302 0000FF93 44656C657465642E2E- <1> Msg_Deleted: db "Deleted...", 0Dh, 0Ah, 0
302 0000FF9C 2E0D0A00 <1>
303                                     <1>
304                                     <1> Msg_Permission_Denied:
305 0000FFA0 07 <1> db 7
306 0000FFA1 5065726D697373696F- <1> db "Permission denied!", 0Dh, 0Ah, 0
306 0000FFAA 6E2064656E69656421- <1>
306 0000FFB3 0D0A00 <1>
307                                     <1>
308                                     <1> ; 04/03/2016
309 0000FFB6 4E657720 <1> Msg_New: db "New "
310 0000FFBA 00 <1> db 0
311                                     <1> Str_Attributes:
312 0000FFBB 417474726962757465- <1> db "Attributes : "
312 0000FFC4 73203A20 <1>
313 0000FFC8 4E4F524D414C <1> Attr_Chars: db "NORMAL"
314 0000FFCE 00 <1> db 0
315                                     <1>
316                                     <1> ; 06/03/2016
317                                     <1> ; CMD_INTR.ASM - 16/11/2010
318                                     <1> Msg_DoYouWantRename:
319 0000FFCF 446F20796F75207761- <1> db "Do you want to rename ", 0
319 0000FFD8 6E7420746F2072656E- <1>
319 0000FFE1 616D652000 <1>
320 0000FFE6 66696C652000 <1> Rename_File: db "file ", 0
321 0000FFEC 6469726563746F7279- <1> Rename_Directory: db "directory ", 0
321 0000FFF5 2000 <1>
322 0000FFF7 00<rept> <1> Rename_OldName: times 13 db 0
323 00010004 20617320 <1> Msg_File_rename_as: db " as "
324 00010008 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 00010015 4E6F742073616D6520- <1> db "Not same drive!"
329 0001001E 647269766521 <1>
330 00010024 0D0A00 <1> db 0Dh, 0Ah, 0
331                                     <1>
332                                     <1> Msg_DoYouWantMoveFile:
333 00010027 446F20796F75207761- <1> db "Do you want to move file", 0
333 00010030 6E7420746F206D6F76- <1>
333 00010039 652066696C6500 <1>
334                                     <1>
335                                     <1> msg_insufficient_disk_space:
336 00010040 496E73756666696369- <1> db "Insufficient disk space!"
336 00010049 656E74206469736B20- <1>
336 00010052 737061636521 <1>
337 00010058 0D0A00 <1> db 0Dh, 0Ah, 0
338                                     <1>
339                                     <1> ; 01/08/2010
340                                     <1> msg_source_file:
341 0001005B 0D0A536F7572636520- <1> db 0Dh, 0Ah, "Source file name : "
341 00010064 66696C65206E616D65- <1>
341 0001006D 2020202020203A2020- <1>
341 00010076 20 <1>
342                                     <1> msg_source_file_drv:
343 00010077 203A00 <1> db " :", 0
344                                     <1> msg_destination_file:
345 0001007A 0D0A44657374696E61- <1> db 0Dh, 0Ah, "Destination file name : "
345 00010083 74696F6E2066696C65- <1>
345 0001008C 206E616D65203A2020- <1>
345 00010095 20 <1>
346                                     <1> msg_destination_file_drv:
347 00010096 203A00 <1> db " :", 0
348                                     <1> msg_copy_nextline:
349 00010099 0D0A00 <1> db 0Dh, 0Ah, 0
350                                     <1>
351                                     <1> ; 15/03/2016
352                                     <1> ; CMD_INTR.ASM
353                                     <1>
354                                     <1> Msg_DoYouWantOverWriteFile:
355 0001009C 446F20796F75207761- <1> db "Do you want to overwrite file ",0
355 000100A5 6E7420746F206F7665- <1>
355 000100AE 727772697465206669- <1>
355 000100B7 6C652000 <1>
356                                     <1>
357                                     <1> Msg_DoYouWantCopyFile:
358 000100BB 446F20796F75207761- <1> db "Do you want to copy file",0
358 000100C4 6E7420746F20636F70- <1>
358 000100CD 792066696C6500 <1>
359                                     <1>
360                                     <1> Msg_read_file_error_before_EOF:
361 000100D4 46696C652072656164- <1> db "File reading error! (before EOF)"
361 000100DD 696E67206572726F72- <1>
```

```

361 000100E6 2120286265666F7265- <1>
361 000100EF 20454F4629          <1>
362 000100F4 0A0A00              <1>                db 0Ah, 0Ah, 0
363                                     <1>
364                                     <1> ; 18/03/2016
365                                     <1> ; TRDOS 386 (v2.0) mainprog copy procedure
366                                     <1> msg_reading:
367 000100F7 52656164696E672E2E- <1>                db "Reading... ", 0
367 00010100 2E2000              <1>
368                                     <1> msg_writing:
369 00010103 57726974696E672E2E- <1>                db "Writing... ", 0
369 0001010C 2E2000              <1>
370                                     <1> percentagestr:
371 0001010F 2020202500          <1>                db "  %", 0 ; "  0%" .. "100%"
372                                     <1> ; 11/04/2016
373                                     <1> Msg_No_Set_Space:
374 00010114 496E73756666696369- <1>                db "Insufficient environment space!"
374 0001011D 656E7420656E766972- <1>
374 00010126 6F6E6D656E74207370- <1>
374 0001012F 61636521            <1>
375 00010133 0D0A00              <1>                db 0Dh, 0Ah, 0
376                                     <1> ; 18/04/2016
377                                     <1> isc_msg:
378 00010136 0D0A                <1>                db 0Dh, 0Ah
379 00010138 494E56414C49442053- <1>                db "INVALID SYSTEM CALL", 0
379 00010141 595354454D2043414C- <1>
379 0001014A 4C00              <1>
380                                     <1> usi_msg:
381 0001014C 0D0A                <1>                db 0Dh, 0Ah
382 0001014E 554E444546494E4544- <1>                db "UNDEFINED SOFTWARE INTERRUPT", 0
382 00010157 20534F465457415245- <1>
382 00010160 20494E544552555550- <1>
382 00010169 5400              <1>
383                                     <1> ifc_msg:
384 0001016B 0D0A                <1>                db 0Dh, 0Ah
385 0001016D 494E56414C49442046- <1>                db "INVALID FUNCTION CALL"
385 00010176 554E4354494F4E2043- <1>
385 0001017F 414C4C            <1>
386                                     <1> inv_msg_for_trdos_v2:
387 00010182 20                  <1>                db 20h
388 00010183 666F72205452444F53- <1>                db "for TRDOS v2!"
388 0001018C 20763221          <1>
389 00010190 07                  <1>                db 07h
390 00010191 0D0A                <1>                db 0Dh, 0Ah
391 00010193 0D0A                <1>                db 0Dh, 0Ah
392 00010195 494E5420          <1>                db "INT "
393 00010199 303068            <1> int_num_str: db "00h"
394 0001019C 0D0A                <1>                db 0Dh, 0Ah
395 0001019E 454158203A20      <1>                db "EAX : "
396 000101A4 303030303030303068- <1> eax_str:      db "00000000h", 0Dh, 0Ah
396 000101AD 0D0A                <1>
397 000101AF 454950203A20      <1>                db "EIP : "
398 000101B5 303030303030303068- <1> eip_str:      db "00000000h", 0Dh, 0Ah, 0
398 000101BE 0D0A00            <1>
399                                     <1>
400                                     <1> ; 07/10/2016
401                                     <1> ; Device names & parameters (for kernel devices)
402                                     <1>
403 000101C1 90                  <1> align 2
404                                     <1> KDEV_NAME:
405 000101C2 5454590000000000    <1>                db 'TTY',0,0,0,0,0 ; 1
406 000101CA 4D454D0000000000    <1>                db 'MEM',0,0,0,0,0 ; 2
407 000101D2 4644300000000000    <1>                db 'FD0',0,0,0,0,0 ; 3
408 000101DA 4644310000000000    <1>                db 'FD1',0,0,0,0,0 ; 4
409 000101E2 4844300000000000    <1>                db 'HD0',0,0,0,0,0 ; 5
410 000101EA 4844310000000000    <1>                db 'HD1',0,0,0,0,0 ; 6
411 000101F2 4844320000000000    <1>                db 'HD2',0,0,0,0,0 ; 7
412 000101FA 4844330000000000    <1>                db 'HD3',0,0,0,0,0 ; 8
413 00010202 4C50540000000000    <1>                db 'LPT',0,0,0,0,0 ; 9
414 0001020A 5454593000000000    <1>                db 'TTY0',0,0,0,0 ; 10
415 00010212 5454593100000000    <1>                db 'TTY1',0,0,0,0 ; 11
416 0001021A 5454593200000000    <1>                db 'TTY2',0,0,0,0 ; 12
417 00010222 5454593300000000    <1>                db 'TTY3',0,0,0,0 ; 13
418 0001022A 5454593400000000    <1>                db 'TTY4',0,0,0,0 ; 14
419 00010232 5454593500000000    <1>                db 'TTY5',0,0,0,0 ; 15
420 0001023A 5454593600000000    <1>                db 'TTY6',0,0,0,0 ; 16
421 00010242 5454593700000000    <1>                db 'TTY7',0,0,0,0 ; 17
422 0001024A 5454593800000000    <1>                db 'TTY8',0,0,0,0 ; 18
423 00010252 5454593900000000    <1>                db 'TTY9',0,0,0,0 ; 19
424 0001025A 434F4D3100000000    <1>                db 'COM1',0,0,0,0 ; 18
425 00010262 434F4D3200000000    <1>                db 'COM2',0,0,0,0 ; 19
426                                     <1>                ;db 'CONSOLE',0 ; 1
427                                     <1>                ;db 'PRINTER',0 ; 9
428                                     <1>                ;db 'CDROM' ; 20
429                                     <1>                ;db 'CDROM0' ; 20
430                                     <1>                ;db 'CDROM1' ; 21
431                                     <1>                ;db 'DVD' ; 22
432                                     <1>                ;db 'DVD0' ; 22
433                                     <1>                ;db 'DVD1' ; 23
434                                     <1>                ;db 'USB' ; 24
435                                     <1>                ;db 'USB0' ; 24
436                                     <1>                ;db 'USB1' ; 25
437                                     <1>                ;db 'USB2' ; 26
438                                     <1>                ;db 'USB3' ; 27
439                                     <1>                ;db 'KEYBOARD' ; 1
440                                     <1>                ;db 'MOUSE' ; 28
441                                     <1>                ;db 'SOUND' ; 29
442                                     <1>                ;db 'VGA',0,0,0,0 ; 30
443                                     <1>                ;db 'CGA',0,0,0,0 ; 31
444                                     <1>                ;db 'AUDIO',0,0,0,0 ; 29
445                                     <1>                ;db 'VIDEO',0,0,0,0 ; 32
446                                     <1>                ;db 'MUSIC',0,0,0,0 ; 33

```



```
447 <1> ;db 'ETHERNET' ; 34
448 <1> ;db 'SD0',0,0,0,0,0 ; 35
449 <1> ;db 'SD1',0,0,0,0,0 ; 36
450 <1> ;db 'SD2',0,0,0,0,0 ; 37
451 <1> ;db 'SD3',0,0,0,0,0 ; 38
452 <1> ;db 'SATA0' ; 35
453 <1> ;db 'SATA1' ; 36
454 <1> ;db 'SATA2' ; 37
455 <1> ;db 'SATA3' ; 38
456 <1> ;db 'PATA0',0,0,0 ; 5
457 <1> ;db 'PATA1',0,0,0 ; 6
458 <1> ;db 'PATA2',0,0,0 ; 7
459 <1> ;db 'PATA3',0,0,0 ; 8
460 <1> ;db 'WIRELESS' ; 39
461 <1> ;db 'HDMI',0,0,0,0 ; 40
462 0001026A 4E554C4C00000000 <1> db 'NULL',0,0,0,0 ; 0
463 <1>
464 <1> NumOfKernelDevNames equ ($-KDEV_NAME) / 8 ; 20 (07/10/2016)
465 <1>
466 <1> KDEV_NUMBER:
467 00010272 010203040506070809 <1> db 1,2,3,4,5,6,7,8,9
468 0001027B 0A0B0C0D0E0F101112- <1> db 10,11,12,13,14,15,16,17,18,19
468 00010284 13 <1>
469 00010285 121300 <1> db 18,19,0
470 <1>
471 <1> NumOfKernelDevices equ $ - KDEV_NUMBER
472 <1>
473 <1> KDEV_OADDR:
474 00010288 [A6FA0000] <1> dd otty ;tty ; 1
475 0001028C [A6FA0000] <1> dd sret ;mem ; 2
476 00010290 [A6FA0000] <1> dd sret ;fd0 ; 3
477 00010294 [A6FA0000] <1> dd sret ;fd1 ; 4
478 00010298 [A6FA0000] <1> dd sret ;hd0 ; 5
479 0001029C [A6FA0000] <1> dd sret ;hd1 ; 6
480 000102A0 [A6FA0000] <1> dd sret ;hd2 ; 7
481 000102A4 [A6FA0000] <1> dd sret ;hd3 ; 8
482 000102A8 [A6FA0000] <1> dd sret ;lpt ; 9
483 000102AC [A6FA0000] <1> dd ocvt ;tty0 ; 10
484 000102B0 [A6FA0000] <1> dd ocvt ;tty1 ; 11
485 000102B4 [A6FA0000] <1> dd ocvt ;tty2 ; 12
486 000102B8 [A6FA0000] <1> dd ocvt ;tty3 ; 13
487 000102BC [A6FA0000] <1> dd ocvt ;tty4 ; 14
488 000102C0 [A6FA0000] <1> dd ocvt ;tty5 ; 15
489 000102C4 [A6FA0000] <1> dd ocvt ;tty6 ; 16
490 000102C8 [A6FA0000] <1> dd ocvt ;tty7 ; 17
491 000102CC [A6FA0000] <1> dd ocvt ;tty8 ; 18
492 000102D0 [A6FA0000] <1> dd ocvt ;tty9 ; 19
493 <1> ;dd ocvt ;com1 ; 18
494 <1> ;dd ocvt ;com2 ; 19
495 000102D4 [A6FA0000] <1> dd sret ;null ; 20
496 <1> KDEV_CADDR:
497 000102D8 [A6FA0000] <1> dd ctty ;tty ; 1
498 000102DC [A6FA0000] <1> dd cret ;mem ; 2
499 000102E0 [A6FA0000] <1> dd cret ;fd0 ; 3
500 000102E4 [A6FA0000] <1> dd cret ;fd1 ; 4
501 000102E8 [A6FA0000] <1> dd cret ;hd0 ; 5
502 000102EC [A6FA0000] <1> dd cret ;hd1 ; 6
503 000102F0 [A6FA0000] <1> dd cret ;hd2 ; 7
504 000102F4 [A6FA0000] <1> dd cret ;hd3 ; 8
505 000102F8 [A6FA0000] <1> dd cret ;lpt ; 9
506 000102FC [A6FA0000] <1> dd ocvt ;tty0 ; 10
507 00010300 [A6FA0000] <1> dd ccvt ;tty1 ; 11
508 00010304 [A6FA0000] <1> dd ccvt ;tty2 ; 12
509 00010308 [A6FA0000] <1> dd ccvt ;tty3 ; 13
510 0001030C [A6FA0000] <1> dd ccvt ;tty4 ; 14
511 00010310 [A6FA0000] <1> dd ccvt ;tty5 ; 15
512 00010314 [A6FA0000] <1> dd ccvt ;tty6 ; 16
513 00010318 [A6FA0000] <1> dd ccvt ;tty7 ; 17
514 0001031C [A6FA0000] <1> dd ccvt ;tty8 ; 18
515 00010320 [A6FA0000] <1> dd ccvt ;tty9 ; 19
516 <1> ;dd ccvt ;com1 ; 18
517 <1> ;dd ccvt ;com2 ; 19
518 00010324 [A6FA0000] <1> dd cret ;null ; 20
519 <1>
520 <1> KDEV_RADDR:
521 00010328 [A6FA0000] <1> dd rtty ;tty ; 1
522 0001032C [A6FA0000] <1> dd rmem ;mem ; 2
523 00010330 [A6FA0000] <1> dd rfd ;fd0 ; 3
524 00010334 [A6FA0000] <1> dd rfd ;fd1 ; 4
525 00010338 [A6FA0000] <1> dd rhd ;hd0 ; 5
526 0001033C [A6FA0000] <1> dd rhd ;hd1 ; 6
527 00010340 [A6FA0000] <1> dd rhd ;hd2 ; 7
528 00010344 [A6FA0000] <1> dd rhd ;hd3 ; 8
529 00010348 [A6FA0000] <1> dd rlpt ;lpt ; 9
530 0001034C [A6FA0000] <1> dd rcvt ;tty0 ; 10
531 00010350 [A6FA0000] <1> dd rcvt ;tty1 ; 11
532 00010354 [A6FA0000] <1> dd rcvt ;tty2 ; 12
533 00010358 [A6FA0000] <1> dd rcvt ;tty3 ; 13
534 0001035C [A6FA0000] <1> dd rcvt ;tty4 ; 14
535 00010360 [A6FA0000] <1> dd rcvt ;tty5 ; 15
536 00010364 [A6FA0000] <1> dd rcvt ;tty6 ; 16
537 00010368 [A6FA0000] <1> dd rcvt ;tty7 ; 17
538 0001036C [A6FA0000] <1> dd rcvt ;tty8 ; 18
539 00010370 [A6FA0000] <1> dd rcvt ;tty9 ; 19
540 <1> ;dd rcvt ;com1 ; 18
541 <1> ;dd rcvt ;com2 ; 19
542 00010374 [A3FA0000] <1> dd rnull ;null ; 20
543 <1> KDEV_WADDR:
544 00010378 [A6FA0000] <1> dd wtty ;tty ; 1
545 0001037C [A6FA0000] <1> dd wmem ;mem ; 2
546 00010380 [A6FA0000] <1> dd wfd ;fd0 ; 3
547 00010384 [A6FA0000] <1> dd wfd ;fd1 ; 4
```

```
548 00010388 [A6FA0000] <1> dd whd ;hd0 ; 5
549 0001038C [A6FA0000] <1> dd whd ;hd1 ; 6
550 00010390 [A6FA0000] <1> dd whd ;hd2 ; 7
551 00010394 [A6FA0000] <1> dd whd ;hd3 ; 8
552 00010398 [A6FA0000] <1> dd wlpt ;lpt ; 9
553 0001039C [A6FA0000] <1> dd xmtt ;tty0 ; 10
554 000103A0 [A6FA0000] <1> dd xmtt ;tty1 ; 11
555 000103A4 [A6FA0000] <1> dd xmtt ;tty2 ; 12
556 000103A8 [A6FA0000] <1> dd xmtt ;tty3 ; 13
557 000103AC [A6FA0000] <1> dd xmtt ;tty4 ; 14
558 000103B0 [A6FA0000] <1> dd xmtt ;tty5 ; 15
559 000103B4 [A6FA0000] <1> dd xmtt ;tty6 ; 16
560 000103B8 [A6FA0000] <1> dd xmtt ;tty7 ; 17
561 000103BC [A6FA0000] <1> dd xmtt ;tty8 ; 18
562 000103C0 [A6FA0000] <1> dd xmtt ;tty9 ; 19
563 <1> ;dd xmtt ;com1 ; 18
564 <1> ;dd xmtt ;com2 ; 19
565 000103C4 [A4FA0000] <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 000103C8 07 <1> db 00000111b; tty, 1
579 000103C9 07 <1> db 00000111b; mem, 2
580 000103CA 8F <1> db 10001111b; fd0, 3
581 000103CB 8F <1> db 10001111b; fd1, 4
582 000103CC 8F <1> db 10001111b; hd0, 5
583 000103CD 8F <1> db 10001111b; hd1, 6
584 000103CE 8F <1> db 10001111b; hd2, 7
585 000103CF 8F <1> db 10001111b; hd3, 8
586 000103D0 07 <1> db 00000111b ; lpt, 9
587 000103D1 07 <1> db 00000111b; tty0, 10
588 000103D2 07 <1> db 00000111b; tty1, 11
589 000103D3 07 <1> db 00000111b; tty2, 12
590 000103D4 07 <1> db 00000111b; tty3, 13
591 000103D5 07 <1> db 00000111b; tty4, 14
592 000103D6 07 <1> db 00000111b; tty5, 15
593 000103D7 07 <1> db 00000111b; tty6, 16
594 000103D8 07 <1> db 00000111b; tty7, 17
595 000103D9 07 <1> db 00000111b; tty8, 18
596 000103DA 07 <1> db 00000111b; tty9, 19
597 <1> ;db 00000111b; com1, 18
598 <1> ;db 00000111b; com2, 19
599 000103DB 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 availability
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 000103DC 0000000010203000400- <1> db 0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0
615 000103E5 05060708090000 <1>
2313
2314 ; 27/08/2014
2315 scr_row:
2316 000103EC E0810B00 dd 0B8000h + 0A0h + 0A0h + 0A0h ; Row 3
2317 scr_col:
2318 000103F0 00000000 dd 0
2319
2320 Align 4
2321 ; 15/04/2016
2322 ; TRDOS 386 (TRDOS v2.0)
2323
2324 ; 21/08/2014
2325 ilist:
2326 ;times 32 dd cpu_except ; INT 0 to INT 1Fh
2327 ;
2328 ; Exception list
2329 ; 25/08/2014
2330 000103F4 [17090000] dd exc0 ; 0h, Divide-by-zero Error
2331 000103F8 [1E090000] dd exc1
2332 000103FC [25090000] dd exc2
2333 00010400 [2C090000] dd exc3
2334 00010404 [30090000] dd exc4
2335 00010408 [34090000] dd exc5
2336 0001040C [38090000] dd exc6 ; 06h, Invalid Opcode
2337 00010410 [3C090000] dd exc7
2338 00010414 [40090000] dd exc8
2339 00010418 [44090000] dd exc9
2340 0001041C [48090000] dd exc10
2341 00010420 [4C090000] dd exc11
2342 00010424 [50090000] dd exc12
2343 00010428 [54090000] dd exc13 ; 0Dh, General Protection Fault
2344 0001042C [58090000] dd exc14 ; 0Eh, Page Fault
2345 00010430 [5C090000] dd exc15
```

```

2346 00010434 [60090000]          dd      exc16
2347 00010438 [64090000]          dd      exc17
2348 0001043C [68090000]          dd      exc18
2349 00010440 [6C090000]          dd      exc19
2350 00010444 [70090000]          dd      exc20
2351 00010448 [74090000]          dd      exc21
2352 0001044C [78090000]          dd      exc22
2353 00010450 [7C090000]          dd      exc23
2354 00010454 [80090000]          dd      exc24
2355 00010458 [84090000]          dd      exc25
2356 0001045C [88090000]          dd      exc26
2357 00010460 [8C090000]          dd      exc27
2358 00010464 [90090000]          dd      exc28
2359 00010468 [94090000]          dd      exc29
2360 0001046C [98090000]          dd      exc30
2361 00010470 [9C090000]          dd      exc31
2362
IRQ_list: ; 28/02/2017 ('syscalbac')
2363          ; Interrupt list
2364 00010474 [8B060000]          dd      timer_int      ; INT 20h
2365          ;dd      irq0
2366 00010478 [FF0D0000]          dd      kb_int        ; 24/01/2016
2367          ;dd      irq1
2368 0001047C [6D080000]          dd      irq2
2369          ; COM2 int
2370 00010480 [71080000]          dd      irq3
2371          ; COM1 int
2372 00010484 [7C080000]          dd      irq4
2373 00010488 [87080000]          dd      irq5
2374
;DISKETTE_INT: ;06/02/2015
2375 0001048C [B0410000]          dd      fdc_int          ; 16/02/2015, IRQ 6 handler
2376          ;dd      irq6
2377
; Default IRQ 7 handler against spurious IRQs (from master PIC)
2378 ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
2379 00010490 [F60B0000]          dd      default_irq7 ; 25/02/2015
2380          ;dd      irq7
2381
; Real Time Clock Interrupt
2382 00010494 [F6070000]          dd      rtc_int          ; 23/02/2015, IRQ 8 handler
2383          ;dd      irq8      ; INT 28h
2384          dd      irq9
2385 0001049C [9B080000]          dd      irq10
2386 000104A0 [9F080000]          dd      irq11
2387 000104A4 [A3080000]          dd      irq12
2388 000104A8 [A7080000]          dd      irq13
2389
;HDISK_INT1: ;06/02/2015
2390 000104AC [2C4B0000]          dd      hdc1_int      ; 21/02/2015, IRQ 14 handler
2391          ;dd      irq14
2392
;HDISK_INT2: ;06/02/2015
2393 000104B0 [534B0000]          dd      hdc2_int      ; 21/02/2015, IRQ 15 handler
2394          ;dd      irq15 ; INT 2Fh
2395          ; 14/08/2015
2396          ;dd      sysent      ; INT 30h (system calls)
2397
2398          ; 15/04/2016
2399          ; TRDOS 386 (TRDOS v2.0) Software Interrupts
2400
2401 000104B4 [11050100]          dd      int30h          ; Reserved for
2402          ; !!! Retro UNIX (RUNIX) !!!
2403          ; !!! SINGLIX !!! System Calls
2404 000104B8 [F2140000]          dd      int31h          ; Video BIOS (IBM PC/AT, Int 10h)
2405 000104BC [1E0C0000]          dd      int32h          ; Keyboard Functions (IBM PC/AT, Int 16h)
2406 000104C0 [66420000]          dd      int33h          ; DISK I/O (IBM PC/AT, Int 13h)
2407 000104C4 [72F30000]          dd      int34h          ; #IOCTL# (I/O port access support for ring 3)
2408 000104C8 [DB580000]          dd      int35h          ; Time/Date Functions (IBM PC/AT, Int 1Ah)
2409 000104CC [AA0A0000]          dd      ignore_int     ; INT 36h : Timer Functions
2410 000104D0 [AA0A0000]          dd      ignore_int     ; INT 37h
2411 000104D4 [AA0A0000]          dd      ignore_int     ; INT 38h
2412 000104D8 [AA0A0000]          dd      ignore_int     ; INT 39h
2413 000104DC [AA0A0000]          dd      ignore_int     ; INT 3Ah
2414 000104E0 [AA0A0000]          dd      ignore_int     ; INT 3Bh
2415 000104E4 [AA0A0000]          dd      ignore_int     ; INT 3Ch
2416 000104E8 [AA0A0000]          dd      ignore_int     ; INT 3Dh
2417 000104EC [AA0A0000]          dd      ignore_int     ; INT 3Eh
2418 000104F0 [AA0A0000]          dd      ignore_int     ; INT 3Fh
2419 000104F4 [C1C20000]          dd      sysent          ; INT 40h : !!! TRDOS 386 System Calls !!!
2420          ;dd      ignore_int
2421 000104F8 00000000          dd      0
2422
2423          ; 20/08/2014
2424          ; /* This is the default interrupt "handler" :-) */
2425          ; Linux v0.12 (head.s)
2426
int_msg:
2427 000104FC 556E6B6E6F776E2069- db "Unknown interrupt ! ", 0
2427 00010505 6E7465727275707420-
2427 0001050E 212000
2428
2429          ; 15/04/2016
2430          ; TRDOS 386 (TRDOS v2.0)
2431
2432          ; 29/04/2016
2433 int30h:
2434 trdos_isc_routine:
2435          ; 02/05/2016
2436          ; 01/05/2016
2437          ; 29/04/2016
2438          ; 18/04/2016
2439          ; 15/04/2016 (TRDOS 386 = TRDOS v2.0)
2440          ; 17/04/2011 (TRDOS v1.0, 'IFC.ASM')
2441          ; 03/02/2011 ('trdos_ifc_routine')
2442          ;
2443 00010511 8B1C24          mov     ebx, [esp] ; EIP (next)
2444 00010514 83EB02          sub     ebx, 2 ; EIP (CD ##h)
2445

```

```
2446 00010517 89C1          mov     ecx, eax
2447 00010519 8A4301        mov     al, [ebx+1] ; CDh ##h
2448
2449 0001051C 66BA1000        mov     dx, KDATA
2450 00010520 8EDA          mov     ds, dx
2451 00010522 8EC2          mov     es, dx
2452
2453 00010524 FC          cld
2454 00010525 8B15[20380100]  mov     edx, [k_page_dir]
2455 0001052B 0F22DA        mov     cr3, edx
2456
2457 0001052E E8962DFFFF      call    bytetohe
2458 00010533 66A3[99010100]  mov     [int_num_str], ax
2459
2460 00010539 89D8          mov     eax, ebx ; EIP
2461 0001053B E8C92DFFFF      call    dwordtohex
2462 00010540 8915[B5010100]  mov     [eip_str], edx
2463 00010546 A3[B9010100]    mov     [eip_str+4], eax
2464
2465 0001054B 89C8          mov     eax, ecx
2466 0001054D E8B72DFFFF      call    dwordtohex
2467 00010552 8915[A4010100]  mov     [eax_str], edx
2468 00010558 A3[A8010100]    mov     [eax_str+4], eax
2469
2470 0001055D 43          inc     ebx
2471 0001055E 8A03          mov     al, [ebx] ; Interrupt number
2472
2473 trdos_isc_handler:
2474 00010560 80FE30        cmp     dh, 30h ; Retro UNIX, SINGLIX System calls
2475 00010563 7507          jne     short trdos_usi_handler
2476 00010565 BE[36010100]  mov     esi, isc_msg
2477 0001056A EB05          jmp     short loc_write_inv_system_call_msg
2478
2479 trdos_usi_handler:
2480 0001056C BE[4C010100]  mov     esi, usi_msg
2481
2482 loc_write_inv_system_call_msg:
2483 00010571 E8475DFFFF      call    print_msg
2484          ; 29/04/2016
2485 00010576 BE[82010100]  mov     esi, inv_msg_for_trdos_v2
2486 0001057B E83D5DFFFF      call    print_msg
2487
2488 loc_ifc_terminate_process:
2489          ; u.uno = process number
2490          ; 29/04/2016
2491
2492          ; 02/05/2016
2493 00010580 FE05[A14E0100]  inc     byte [sysflg] ; 0FFh -> 0
2494
2495 00010586 B801000000      mov     eax, 1
2496 0001058B E906C0FFFF      jmp     sysexit
2497
2498          ; 07/03/2015
2499          ; Temporary Code
2500 display_disks:
2501 00010590 803D[565C0000]00  cmp     byte [fd0_type], 0
2502 00010597 7605          jna     short ddsks1
2503 00010599 E87D000000      call    pdskm
2504
2505 ddsks1:
2506 0001059E 803D[575C0000]00  cmp     byte [fd1_type], 0
2507 000105A5 760C          jna     short ddsks2
2508 000105A7 C605[EB060100]31  mov     byte [dskx], '1'
2509 000105AE E868000000      call    pdskm
2510
2511 ddsks2:
2512 000105B3 803D[585C0000]00  cmp     byte [hd0_type], 0
2513 000105BA 7654          jna     short ddsks6
2514 000105BC 66C705[E9060100]68-  mov     word [dsktype], 'hd'
2515 000105C4 64
2516 000105C5 C605[EB060100]30  mov     byte [dskx], '0'
2517 000105CC E84A000000      call    pdskm
2518
2519 ddsks3:
2520 000105D1 803D[595C0000]00  cmp     byte [hd1_type], 0
2521 000105D8 7636          jna     short ddsks6
2522 000105DA C605[EB060100]31  mov     byte [dskx], '1'
2523 000105E1 E835000000      call    pdskm
2524
2525 ddsks4:
2526 000105E6 803D[5A5C0000]00  cmp     byte [hd2_type], 0
2527 000105ED 7621          jna     short ddsks6
2528 000105EF C605[EB060100]32  mov     byte [dskx], '2'
2529 000105F6 E820000000      call    pdskm
2530
2531 ddsks5:
2532 000105FB 803D[5B5C0000]00  cmp     byte [hd3_type], 0
2533 00010602 760C          jna     short ddsks6
2534 00010604 C605[EB060100]33  mov     byte [dskx], '3'
2535 0001060B E80B000000      call    pdskm
2536
2537 ddsks6:
2538 00010610 BE[13070100]  mov     esi, nextline
2539 00010615 E806000000      call    pdskml
2540
2541 pdskm_ok:
2542          retn
2543
2544 pdskm:
2545          mov     esi, dsk_ready_msg
2546
2547 pdskml:
2548          lodsb
2549          or      al, al
2550          jz      short pdskm_ok
2551          push    esi
2552          ; 13/05/2016
2553          mov     ebx, 7 ; Black background,
2554          ; light gray forecolor
2555          ; Video page 0 (bh=0)
2556          call    _write_tty
```



```

2547 00010630 5E                pop     esi
2548 00010631 EBED              jmp     short pdskml
2549
2550 00010633 90                Align 2
2551                                ; 21/08/2014
2552 exc_msg:
2553 00010634 435055206578636570-    db "CPU exception ! "
2553 0001063D 74696F6E202120
2554
2555 00010644 3F3F68202045495020-    excnstr:                ; 25/08/2014
2555 0001064D 3A20                db "??h", "   EIP : "
2556
2557 0001064F 00<rept>          EIPstr: ; 29/08/2014
2558                                times 12 db 0
2559
2560                                ; 23/02/2015
2561                                ; 25/08/2014
2562 ;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 0001065B 07                db      07h
2570 0001065C 0D0A                db      0Dh, 0Ah
2571                                ;db      "MEMORY ALLOCATION INFO", 0Dh, 0Ah, 0Dh, 0Ah
2572 0001065E 546F74616C206D656D-    db      "Total memory : "
2572 00010667 6F7279203A20
2573
2574 0001066D 30303030303030303030-    mem_total_b_str: ; 10 digits
2574 00010676 302062797465730D0A    db      "0000000000 bytes", 0Dh, 0Ah
2575 0001067F 202020202020202020-    db      "
2575 00010688 202020202020202020                ", 20h, 20h, 20h
2576
2577 00010691 30303030303030302070-    mem_total_p_str: ; 7 digits
2577 0001069A 616765730D0A    db      "0000000 pages", 0Dh, 0Ah
2578 000106A0 0D0A                db      0Dh, 0Ah
2579 000106A2 46726565206D656D6F-    db      "Free memory : "
2579 000106AB 727920203A20
2580
2581 000106B1 3F3F3F3F3F3F3F3F3F3F-    free_mem_b_str:  ; 10 digits
2581 000106BA 3F2062797465730D0A    db      "?????????? bytes", 0Dh, 0Ah
2582 000106C3 202020202020202020-    db      "
2582 000106CC 202020202020202020                ", 20h, 20h, 20h
2583
2584 000106D5 3F3F3F3F3F3F3F3F2070-    free_mem_p_str:  ; 7 digits
2584 000106DE 616765730D0A    db      "??????? pages", 0Dh, 0Ah
2585 000106E4 0D0A00                db      0Dh, 0Ah, 0
2586
2587
2588 000106E7 0D0A                dsk_ready_msg:
2589                                db      0Dh, 0Ah
2590 000106E9 6664                dsktype:
2591                                db      'fd'
2592 000106EB 30                dskx:
2593 000106EC 20                db      '0'
2594 000106ED 697320524541445920-    db      20h
2594 000106F6 2E2E2E                db      'is READY ...'
2595 000106F9 00                db      0
2596
2597
2598 000106FA 0D0A                setup_error_msg:
2599 000106FC 4469736B2053657475-    db 0Dh, 0Ah
2599 00010705 70204572726F722021    db 'Disk Setup Error !'
2600 0001070E 0D0A00                db 0Dh, 0Ah,0
2601
2602
2603 00010711 0D0A                next2line: ; 08/02/2016
2604                                db      0Dh, 0Ah
2605 00010713 0D0A00                nextline:
2606                                db      0Dh, 0Ah, 0
2607
2608                                ; KERNEL - SYSINIT Messages
2609                                ; 24/08/2015
2610                                ; 13/04/2015 - (Retro UNIX 386 v1 Beginning)
2611                                ; 14/07/2013
2612 ;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 - 26/03/2017 (v2.0.0)'
2623 ;      db 0Dh, 0Ah, 0
2624
2625
2626 00010716 0D0A07                panic_msg:
2627 00010719 4552524F523A204B65-    db 0Dh, 0Ah, 07h
2627 00010722 726E656C2050616E69-    db 'ERROR: Kernel Panic !'
2627 0001072B 632021
2628 0001072E 0D0A00                db 0Dh, 0Ah, 0
2629
2630
2631 ;msg1_drv_not_ready:
2632 ;      db 07h, 0Dh, 0Ah
2633 ;      db 'Drive not ready or read error !'
2634 ;      db 0Dh, 0Ah, 0

```

```
2635      starting_msg:
2636      db "Turkish Rational DOS v2.0 [26/03/2017] ...", 0
2636 0001073A 6174696F6E616C2044-
2636 00010743 4F532076322E30205B-
2636 0001074C 32362F30332F323031-
2636 00010755 375D202E2E2E00
2637      NextLine:
2638 0001075C 0D0A00      db 0Dh, 0Ah, 0
2639
2640 0001075F 90      align 4
2641
2642      %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 00010760 00000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
28 00010769 0000000000000000 <1>
29 00010770 0000000000000000002A- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
29 00010779 2A2A2A2A2A2A2A2A <1>
30 00010780 2A2A2A2A2A2A2A2A2A- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
30 00010789 2A2A2A2A2A2A2A2A <1>
31 00010790 2A2A2A2A2A2A2A2A2A- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
31 00010799 2A2A2A2A2A2A2A2A <1>
32 000107A0 2A2A2A2A2A2A2A2A3F- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
32 000107A9 3F3F3F3F3F3F3F3F <1>
33 000107B0 3F3F3F3F3F3F3F3F3F3F- <1> db 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
33 000107B9 3F3F3F3F3F3F3F3F <1>
34 000107C0 00000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
34 000107C9 0000000000000000 <1>
35 000107D0 0000000000000000002A- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
35 000107D9 2A2A2A2A2A2A2A2A <1>
36 000107E0 2A2A2A2A2A2A2A2A2A- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
36 000107E9 2A2A2A2A2A2A2A2A <1>
37 000107F0 2A2A2A2A2A2A2A2A2A- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah
37 000107F9 2A2A2A2A2A2A2A2A <1>
38 00010800 2A2A2A2A2A2A2A2A2A3F- <1> db 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 02ah, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
38 00010809 3F3F3F3F3F3F3F3F <1>
39 00010810 3F3F3F3F3F3F3F3F3F3F- <1> db 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh, 03fh
39 00010819 3F3F3F3F3F3F3F3F <1>
40     <1> palette1:
41 00010820 00000000002A002A00- <1> db 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
41 00010829 002A2A2A000002A <1>
42 00010830 002A2A15002A2A2A00- <1> db 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah
42 00010839 000000002A0002A <1>
43 00010840 00002A2A2A00002A00- <1> db 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah, 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah
43 00010849 2A2A15002A2A2A <1>
44 00010850 15151515153F153F15- <1> db 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh, 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh
44 00010859 153F3F3F15153F <1>
45 00010860 153F3F3F153F3F3F15- <1> db 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
45 00010869 151515153F153F <1>
46 00010870 15153F3F3F15153F15- <1> db 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
46 00010879 3F3F3F153F3F3F <1>
47 00010880 00000000002A002A00- <1> db 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
47 00010889 002A2A2A000002A <1>
48 00010890 002A2A15002A2A2A00- <1> db 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah
48 00010899 000000002A0002A <1>
49 000108A0 00002A2A2A00002A00- <1> db 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah, 000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah
49 000108A9 2A2A15002A2A2A <1>
50 000108B0 15151515153F153F15- <1> db 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh, 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh
50 000108B9 153F3F3F15153F <1>
51 000108C0 153F3F3F153F3F3F15- <1> db 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
51 000108C9 151515153F153F <1>
52 000108D0 15153F3F3F15153F15- <1> db 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
52 000108D9 3F3F3F153F3F3F <1>
53     <1> palette2:
54 000108E0 00000000002A002A00- <1> db 000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
54 000108E9 002A2A2A000002A <1>
55 000108F0 002A2A2A002A2A2A00- <1> db 000h, 02ah, 02ah, 02ah, 000h, 02ah, 02ah, 02ah, 000h, 000h, 015h, 000h, 000h, 03fh, 000h, 02ah
55 000108F9 001500003F0002A <1>
56 00010900 15002A3F2A00152A00- <1> db 015h, 000h, 02ah, 03fh, 02ah, 000h, 015h, 02ah, 000h, 03fh, 02ah, 02ah, 015h, 02ah, 02ah, 03fh
56 00010909 3F2A2A152A2A3F <1>
57 00010910 00150000152A003F00- <1> db 000h, 015h, 000h, 000h, 015h, 02ah, 000h, 03fh, 000h, 000h, 03fh, 02ah, 02ah, 015h, 000h, 02ah
57 00010919 003F2A2A15002A <1>
58 00010920 152A2A3F002A3F2A00- <1> db 015h, 02ah, 02ah, 03fh, 000h, 02ah, 03fh, 02ah, 000h, 015h, 015h, 000h, 015h, 03fh, 000h, 03fh
58 00010929 151500153F003F <1>
59 00010930 15003F3F2A15152A15- <1> db 015h, 000h, 03fh, 03fh, 02ah, 015h, 015h, 02ah, 015h, 03fh, 02ah, 03fh, 015h, 02ah, 03fh, 03fh
59 00010939 3F2A3F152A3F3F <1>
60 00010940 15000015002A152A00- <1> db 015h, 000h, 000h, 015h, 000h, 02ah, 015h, 02ah, 000h, 015h, 02ah, 02ah, 03fh, 000h, 000h, 03fh
60 00010949 152A2A3F000003F <1>
61 00010950 002A3F2A003F2A2A15- <1> db 000h, 02ah, 03fh, 02ah, 000h, 03fh, 02ah, 02ah, 015h, 000h, 015h, 015h, 000h, 03fh, 015h, 02ah
61 00010959 001515003F152A <1>
62 00010960 15152A3F3F00153F00- <1> db 015h, 015h, 02ah, 03fh, 03fh, 000h, 015h, 03fh, 000h, 03fh, 03fh, 02ah, 015h, 03fh, 02ah, 03fh
62 00010969 3F3F2A153F2A3F <1>
63 00010970 15150015152A153F00- <1> db 015h, 015h, 000h, 015h, 015h, 02ah, 015h, 03fh, 000h, 015h, 03fh, 02ah, 03fh, 015h, 000h, 03fh
63 00010979 153F2A3F15003F <1>
64 00010980 152A3F3F003F3F2A15- <1> db 015h, 02ah, 03fh, 03fh, 000h, 03fh, 03fh, 02ah, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
64 00010989 151515153F153F <1>
65 00010990 15153F3F3F15153F15- <1> db 015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
65 00010999 3F3F3F153F3F3F <1>
66     <1> palette3:
```

67	000109A0	000000000002A002A00-	<1>	db	000h, 000h, 000h, 000h, 000h, 02ah, 000h, 02ah, 000h, 000h, 02ah, 02ah, 02ah, 000h, 000h, 02ah
67	000109A9	002A2A2A000002A	<1>		
68	000109B0	002A2A15002A2A2A15-	<1>	db	000h, 02ah, 02ah, 015h, 000h, 02ah, 02ah, 02ah, 015h, 015h, 015h, 015h, 015h, 03fh, 015h, 03fh
68	000109B9	151515153F153F	<1>		
69	000109C0	15153F3F3F15153F15-	<1>	db	015h, 015h, 03fh, 03fh, 03fh, 015h, 015h, 03fh, 015h, 03fh, 03fh, 03fh, 015h, 03fh, 03fh, 03fh
69	000109C9	3F3F3F153F3F3F	<1>		
70	000109D0	000000050505080808-	<1>	db	000h, 000h, 000h, 005h, 005h, 005h, 008h, 008h, 008h, 00bh, 00bh, 00bh, 00eh, 00eh, 00eh, 011h
70	000109D9	0B0B0B0E0E0E11	<1>		
71	000109E0	11111414141818181C-	<1>	db	011h, 011h, 014h, 014h, 014h, 018h, 018h, 018h, 01ch, 01ch, 01ch, 020h, 020h, 020h, 024h, 024h
71	000109E9	1C1C2020202424	<1>		
72	000109F0	242828282D2D2D3232-	<1>	db	024h, 028h, 028h, 028h, 02dh, 02dh, 02dh, 032h, 032h, 032h, 038h, 038h, 038h, 03fh, 03fh, 03fh
72	000109F9	323838383F3F3F	<1>		
73	00010A00	00003F10003F1F003F-	<1>	db	000h, 000h, 03fh, 010h, 000h, 03fh, 01fh, 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h, 03fh, 03fh
73	00010A09	2F003F3F003F3F	<1>		
74	00010A10	002F3F001F3F00103F-	<1>	db	000h, 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh
74	00010A19	00003F10003F1F	<1>		
75	00010A20	003F2F003F3F002F3F-	<1>	db	000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h, 02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 000h
75	00010A29	001F3F00103F00	<1>		
76	00010A30	003F00003F10003F1F-	<1>	db	000h, 03fh, 000h, 000h, 03fh, 010h, 000h, 03fh, 01fh, 000h, 03fh, 02fh, 000h, 03fh, 03fh, 000h
76	00010A39	003F2F003F3F00	<1>		
77	00010A40	2F3F001F3F00103F1F-	<1>	db	02fh, 03fh, 000h, 01fh, 03fh, 000h, 010h, 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh, 02fh, 01fh
77	00010A49	1F3F271F3F2F1F	<1>		
78	00010A50	3F371F3F3F1F3F3F1F-	<1>	db	03fh, 037h, 01fh, 03fh, 03fh, 01fh, 03fh, 03fh, 01fh, 037h, 03fh, 01fh, 02fh, 03fh, 01fh, 027h
78	00010A59	373F1F2F3F1F27	<1>		
79	00010A60	3F1F1F3F271F3F2F1F-	<1>	db	03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh, 02fh, 01fh, 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 037h
79	00010A69	3F371F3F3F1F37	<1>		
80	00010A70	3F1F2F3F1F273F1F1F-	<1>	db	03fh, 01fh, 02fh, 03fh, 01fh, 027h, 03fh, 01fh, 01fh, 03fh, 01fh, 01fh, 03fh, 027h, 01fh, 03fh
80	00010A79	3F1F1F3F271F3F	<1>		
81	00010A80	2F1F3F371F3F3F1F37-	<1>	db	02fh, 01fh, 03fh, 037h, 01fh, 03fh, 03fh, 01fh, 037h, 03fh, 01fh, 02fh, 03fh, 01fh, 027h, 03fh
81	00010A89	3F1F2F3F1F273F	<1>		
82	00010A90	2D2D3F312D3F362D3F-	<1>	db	02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h, 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh, 03fh, 03fh
82	00010A99	3A2D3F3F2D3F3F	<1>		
83	00010AA0	2D3A3F2D363F2D313F-	<1>	db	02dh, 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h
83	00010AA9	2D2D3F312D3F36	<1>		
84	00010AB0	2D3F3A2D3F3F2D3A3F-	<1>	db	02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh, 03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 02dh
84	00010AB9	2D363F2D313F2D	<1>		
85	00010AC0	2D3F2D2D3F312D3F36-	<1>	db	02dh, 03fh, 02dh, 02dh, 03fh, 031h, 02dh, 03fh, 036h, 02dh, 03fh, 03ah, 02dh, 03fh, 03fh, 02dh
85	00010AC9	2D3F3A2D3F3F2D	<1>		
86	00010AD0	3A3F2D363F2D313F00-	<1>	db	03ah, 03fh, 02dh, 036h, 03fh, 02dh, 031h, 03fh, 000h, 000h, 01ch, 007h, 000h, 01ch, 00eh, 000h
86	00010AD9	001C07001C0E00	<1>		
87	00010AE0	1C15001C1C001C1C00-	<1>	db	01ch, 015h, 000h, 01ch, 01ch, 000h, 01ch, 01ch, 000h, 015h, 01ch, 000h, 00eh, 01ch, 000h, 007h
87	00010AE9	151C000E1C0007	<1>		
88	00010AF0	1C00001C07001C0E00-	<1>	db	01ch, 000h, 000h, 01ch, 007h, 000h, 01ch, 00eh, 000h, 01ch, 015h, 000h, 01ch, 01ch, 000h, 015h
88	00010AF9	1C15001C1C0015	<1>		
89	00010B00	1C000E1C00071C0000-	<1>	db	01ch, 000h, 00eh, 01ch, 000h, 007h, 01ch, 000h, 000h, 01ch, 000h, 000h, 01ch, 007h, 000h, 01ch
89	00010B09	1C00001C07001C	<1>		
90	00010B10	0E001C15001C1C0015-	<1>	db	00eh, 000h, 01ch, 015h, 000h, 01ch, 01ch, 000h, 015h, 01ch, 000h, 00eh, 01ch, 000h, 007h, 01ch
90	00010B19	1C000E1C00071C	<1>		
91	00010B20	0E0E1C110E1C150E1C-	<1>	db	00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 01ch, 01ch
91	00010B29	180E1C1C0E1C1C	<1>		
92	00010B30	0E181C0E151C0E111C-	<1>	db	00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h
92	00010B39	0E0E1C110E1C15	<1>		
93	00010B40	0E1C180E1C1C0E181C-	<1>	db	00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh
93	00010B49	0E151C0E111C0E	<1>		
94	00010B50	0E1C0E0E1C110E1C15-	<1>	db	00eh, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh
94	00010B59	0E1C180E1C1C0E	<1>		
95	00010B60	181C0E151C0E111C14-	<1>	db	018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h
95	00010B69	141C16141C1814	<1>		
96	00010B70	1C1A141C1C141C1C14-	<1>	db	01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h
96	00010B79	1A1C14181C1416	<1>		
97	00010B80	1C14141C16141C1814-	<1>	db	01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah
97	00010B89	1C1A141C1C141A	<1>		
98	00010B90	1C14181C14161C1414-	<1>	db	01ch, 014h, 018h, 01ch, 014h, 016h, 01ch, 014h, 014h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch
98	00010B99	1C14141C16141C	<1>		
99	00010BA0	18141C1A141C1C141A-	<1>	db	018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h, 01ch
99	00010BA9	1C14181C14161C	<1>		
100	00010BB0	000010040010080010-	<1>	db	000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h, 010h, 010h
100	00010BB9	0C001010001010	<1>		
101	00010BC0	000C10000810000410-	<1>	db	000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h
101	00010BC9	00001004001008	<1>		
102	00010BD0	00100C001010000C10-	<1>	db	000h, 010h, 00ch, 000h, 010h, 010h, 000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h
102	00010BD9	00081000041000	<1>		
103	00010BE0	001000001004001008-	<1>	db	000h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h
103	00010BE9	00100C00101000	<1>		
104	00010BF0	0C1000081000041008-	<1>	db	00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h
104	00010BF9	08100A08100C08	<1>		
105	00010C00	100E08101008101008-	<1>	db	010h, 00eh, 008h, 010h, 010h, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah
105	00010C09	0E10080C10080A	<1>		
106	00010C10	100808100A08100C08-	<1>	db	010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh
106	00010C19	100E081010080E	<1>		
107	00010C20	10080C10080A100808-	<1>	db	010h, 008h, 00ch, 010h, 008h, 00ah, 010h, 008h, 008h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h
107	00010C29	100808100A0810	<1>		
108	00010C30	0C08100E081010080E-	<1>	db	00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah, 010h
108	00010C39	10080C10080A10	<1>		
109	00010C40	0B0B100C0B100D0B10-	<1>	db	00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 010h, 010h
109	00010C49	0F0B10100B1010	<1>		
110	00010C50	0B0F100B0D100B0C10-	<1>	db	00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh
110	00010C59	0B0B100C0B100D	<1>		
111	00010C60	0B100F0B10100B0F10-	<1>	db	00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh
111	00010C69	0B0D100B0C100B	<1>		
112	00010C70	0B100B0B100C0B100D-	<1>	db	00bh, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh
112	00010C79	0B100F0B10100B	<1>		
113	00010C80	0F100B0D100B0C1000-	<1>	db	00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
113	00010C89	0000000000000000	<1>		
114	00010C90	000000000000000000-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
114	00010C99	0000000000000000	<1>		
115			<1>		
116			<1>		
117			<1>		
117			<1>		; 04/07/2016
118			<1>		; FONT DATA
119			<1>		
120			<1>		CRT_CHAR_GEN:
121			<1>		vgafont8:
122	00010CA0	00000000000000007E-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 081h, 0a5h, 081h, 0bdh, 099h, 081h, 07eh
122	00010CA9	81A581BD99817E	<1>		
123	00010CB0	7EFFDBFFC3E7FF7E6C-	<1>	db	07eh, 0ffh, 0dbh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 06ch, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h
123	00010CB9	FEFEFE7C381000	<1>		
124	00010CC0	10387CFE7C38100038-	<1>	db	010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 038h, 07ch, 038h, 0feh, 0feh, 07ch, 038h, 07ch
124	00010CC9	7C38FEFE7C387C	<1>		
125	00010CD0	1010387CFE7C387C00-	<1>	db	010h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 07ch, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h
125	00010CD9	00183C3C180000	<1>		
126	00010CE0	FFFFE7C3C3E7FFFF00-	<1>	db	0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h
126	00010CE9	3C664242663C00	<1>		
127	00010CF0	FFC399BDBD99C3FF0F-	<1>	db	0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 00fh, 007h, 00fh, 07dh, 0cch, 0cch, 0cch, 078h
127	00010CF9	070F7DCCCCC78	<1>		

128	00010D00	3C6666663C187E183F-	<1>	db	03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 03fh, 033h, 03fh, 030h, 030h, 070h, 0f0h, 0e0h
128	00010D09	333F303070F0E0	<1>		
129	00010D10	7F637F636367E6C099-	<1>	db	07fh, 063h, 07fh, 063h, 063h, 067h, 0e6h, 0c0h, 099h, 05ah, 03ch, 0e7h, 0e7h, 03ch, 05ah, 099h
129	00010D19	5A3CE7E73C5A99	<1>		
130	00010D20	80E0F8FEF8E0800002-	<1>	db	080h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 080h, 000h, 002h, 00eh, 03eh, 0feh, 03eh, 00eh, 002h, 000h
130	00010D29	0E3EFE3E0E0200	<1>		
131	00010D30	183C7E18187E3C1866-	<1>	db	018h, 03ch, 07eh, 018h, 018h, 07eh, 03ch, 018h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 000h
131	00010D39	66666666006600	<1>		
132	00010D40	7FDBDB7B1B1B1B003E-	<1>	db	07fh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 000h, 03eh, 063h, 038h, 06ch, 06ch, 038h, 0cch, 078h
132	00010D49	63386C6C38CC78	<1>		
133	00010D50	000000007E7E7E0018-	<1>	db	000h, 000h, 000h, 000h, 07eh, 07eh, 07eh, 000h, 018h, 03ch, 07eh, 018h, 07eh, 03ch, 018h, 0ffh
133	00010D59	3C7E187E3C18FF	<1>		
134	00010D60	183C7E181818180018-	<1>	db	018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h
134	00010D69	1818187E3C1800	<1>		
135	00010D70	00180CFE0C18000000-	<1>	db	000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h
135	00010D79	3060FE60300000	<1>		
136	00010D80	0000C0C0C0FE000000-	<1>	db	000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h
136	00010D89	2466FF66240000	<1>		
137	00010D90	00183C7EFFFF000000-	<1>	db	000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 000h, 000h, 000h, 0ffh, 0ffh, 07eh, 03ch, 018h, 000h, 000h
137	00010D99	FFFF7E3C180000	<1>		
138	00010DA0	000000000000000030-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 078h, 078h, 030h, 030h, 000h, 030h, 000h
138	00010DA9	78783030003000	<1>		
139	00010DB0	6C6C6C00000000006C-	<1>	db	06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 0feh, 06ch, 06ch, 000h
139	00010DB9	6CFE6CFE6C6C00	<1>		
140	00010DC0	307CC0780CF8300000-	<1>	db	030h, 07ch, 0c0h, 078h, 00ch, 0f8h, 030h, 000h, 000h, 0c6h, 0cch, 018h, 030h, 066h, 0c6h, 000h
140	00010DC9	C6CC183066C600	<1>		
141	00010DD0	386C3876DCCC760060-	<1>	db	038h, 06ch, 038h, 076h, 0dch, 0cch, 076h, 000h, 060h, 060h, 0c0h, 000h, 000h, 000h, 000h, 000h
141	00010DD9	60C0000000000000	<1>		
142	00010DE0	183060606030180060-	<1>	db	018h, 030h, 060h, 060h, 060h, 030h, 018h, 000h, 060h, 030h, 018h, 018h, 018h, 030h, 060h, 000h
142	00010DE9	30181818306000	<1>		
143	00010DF0	00663CFF3C66000000-	<1>	db	000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 000h
143	00010DF9	3030FC30300000	<1>		
144	00010E00	000000000030306000-	<1>	db	000h, 000h, 000h, 000h, 000h, 030h, 030h, 060h, 000h, 000h, 000h, 0fch, 000h, 000h, 000h, 000h
144	00010E09	0000FC0000000000	<1>		
145	00010E10	000000000030300006-	<1>	db	000h, 000h, 000h, 000h, 000h, 030h, 030h, 000h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h
145	00010E19	0C183060C08000	<1>		
146	00010E20	7CC6CEDEF6E67C0030-	<1>	db	07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 07ch, 000h, 030h, 070h, 030h, 030h, 030h, 030h, 0fch, 000h
146	00010E29	7030303030FC00	<1>		
147	00010E30	78CC0C3860CCFC0078-	<1>	db	078h, 0cch, 00ch, 038h, 060h, 0cch, 0fch, 000h, 078h, 0cch, 00ch, 038h, 00ch, 0cch, 078h, 000h
147	00010E39	CC0C380CCC7800	<1>		
148	00010E40	1C3C6CCCFE0C1E00FC-	<1>	db	01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 01eh, 000h, 0fch, 0c0h, 0f8h, 00ch, 00ch, 0cch, 078h, 000h
148	00010E49	C0F80C0CCC7800	<1>		
149	00010E50	3860C0F8CCCC7800FC-	<1>	db	038h, 060h, 0c0h, 0f8h, 0cch, 0cch, 078h, 000h, 0fch, 0cch, 00ch, 018h, 030h, 030h, 030h, 000h
149	00010E59	CC0C1830303000	<1>		
150	00010E60	78CCCC78CCCC780078-	<1>	db	078h, 0cch, 0cch, 078h, 0cch, 0cch, 078h, 000h, 078h, 0cch, 0cch, 07ch, 00ch, 018h, 070h, 000h
150	00010E69	CCCC7C0C187000	<1>		
151	00010E70	003030000030300000-	<1>	db	000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 060h
151	00010E79	30300000303060	<1>		
152	00010E80	183060C06030180000-	<1>	db	018h, 030h, 060h, 0c0h, 060h, 030h, 018h, 000h, 000h, 000h, 0fch, 000h, 000h, 0fch, 000h, 000h
152	00010E89	00FC0000FC0000	<1>		
153	00010E90	6030180C1830600078-	<1>	db	060h, 030h, 018h, 00ch, 018h, 030h, 060h, 000h, 078h, 0cch, 00ch, 018h, 030h, 000h, 030h, 000h
153	00010E99	CC0C1830003000	<1>		
154	00010EA0	7CC6DEDEDEC0780030-	<1>	db	07ch, 0c6h, 0deh, 0deh, 0deh, 0c0h, 078h, 000h, 030h, 078h, 0cch, 0cch, 0fch, 0cch, 0cch, 000h
154	00010EA9	78CCCCFCCCCC00	<1>		
155	00010EB0	FC66667C6666FC003C-	<1>	db	0fch, 066h, 066h, 07ch, 066h, 066h, 0fch, 000h, 03ch, 066h, 0c0h, 0c0h, 0c0h, 066h, 03ch, 000h
155	00010EB9	66C0C0C0663C00	<1>		
156	00010EC0	F86C6666666CF800FE-	<1>	db	0f8h, 06ch, 066h, 066h, 066h, 06ch, 0f8h, 000h, 0feh, 062h, 068h, 078h, 068h, 062h, 0feh, 000h
156	00010EC9	6268786862FE00	<1>		
157	00010ED0	FE6268786860F0003C-	<1>	db	0feh, 062h, 068h, 078h, 068h, 060h, 0f0h, 000h, 03ch, 066h, 0c0h, 0c0h, 0ceh, 066h, 03eh, 000h
157	00010ED9	66C0C0CE663E00	<1>		
158	00010EE0	CCCCCFCCCCCCC0078-	<1>	db	0cch, 0cch, 0cch, 0fch, 0cch, 0cch, 0cch, 000h, 078h, 030h, 030h, 030h, 030h, 030h, 078h, 000h
158	00010EE9	30303030307800	<1>		
159	00010EF0	1E0C0C0CCCCC7800E6-	<1>	db	01eh, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 0e6h, 066h, 06ch, 078h, 06ch, 066h, 0e6h, 000h
159	00010EF9	666C786C66E600	<1>		
160	00010F00	F06060606266FE00C6-	<1>	db	0f0h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 000h
160	00010F09	EEFEFED6C6C600	<1>		
161	00010F10	C6E6F6DECEC6C60038-	<1>	db	0c6h, 0e6h, 0f6h, 0deh, 0ceh, 0c6h, 0c6h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h
161	00010F19	6CC6C6C66C3800	<1>		
162	00010F20	FC66667C6060F00078-	<1>	db	0fch, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 078h, 0cch, 0cch, 0cch, 0dch, 078h, 01ch, 000h
162	00010F29	CCCCCDC781C00	<1>		
163	00010F30	FC66667C6C66E60078-	<1>	db	0fch, 066h, 066h, 07ch, 06ch, 066h, 0e6h, 000h, 078h, 0cch, 0e0h, 070h, 01ch, 0cch, 078h, 000h
163	00010F39	CCE0701CCC7800	<1>		
164	00010F40	FCB4303030307800CC-	<1>	db	0fch, 0b4h, 030h, 030h, 030h, 030h, 078h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 0fch, 000h
164	00010F49	CCCCCCCCCF00	<1>		
165	00010F50	CCCCCCCCC783000C6-	<1>	db	0cch, 0cch, 0cch, 0cch, 0cch, 078h, 030h, 000h, 0c6h, 0c6h, 0c6h, 0d6h, 0feh, 0eeh, 0c6h, 000h
165	00010F59	C6C6D6FEEEC600	<1>		
166	00010F60	C6C66C38386CC600CC-	<1>	db	0c6h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 030h, 078h, 000h
166	00010F69	CCCC7830307800	<1>		
167	00010F70	FEC68C183266FE0078-	<1>	db	0feh, 0c6h, 08ch, 018h, 032h, 066h, 0feh, 000h, 078h, 060h, 060h, 060h, 060h, 060h, 078h, 000h
167	00010F79	60606060607800	<1>		
168	00010F80	C06030180C06020078-	<1>	db	0c0h, 060h, 030h, 018h, 00ch, 006h, 002h, 000h, 078h, 018h, 018h, 018h, 018h, 018h, 078h, 000h
168	00010F89	18181818187800	<1>		
169	00010F90	10386CC60000000000-	<1>	db	010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
169	00010F99	000000000000FF	<1>		
170	00010FA0	303018000000000000-	<1>	db	030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 076h, 000h
170	00010FA9	00780C7CCC7600	<1>		
171	00010FB0	E060607C6666DC0000-	<1>	db	0e0h, 060h, 060h, 07ch, 066h, 066h, 0dch, 000h, 000h, 000h, 078h, 0cch, 0c0h, 0cch, 078h, 000h
171	00010FB9	0078CCCCC7800	<1>		
172	00010FC0	1C0C0C7CCCC760000-	<1>	db	01ch, 00ch, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
172	00010FC9	0078CCFCC07800	<1>		
173	00010FD0	386C60F06060F00000-	<1>	db	038h, 06ch, 060h, 0f0h, 060h, 060h, 0f0h, 000h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 0f8h
173	00010FD9	0076CCCC7C0CF8	<1>		
174	00010FE0	E0606C766666E60030-	<1>	db	0e0h, 060h, 06ch, 076h, 066h, 066h, 0e6h, 000h, 030h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
174	00010FE9	00703030307800	<1>		
175	00010FF0	0C000C0C0CCCCC78E0-	<1>	db	00ch, 000h, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 0e0h, 060h, 066h, 06ch, 078h, 06ch, 0e6h, 000h
175	00010FF9	60666C786CE600	<1>		
176	00011000	703030303030780000-	<1>	db	070h, 030h, 030h, 030h, 030h, 030h, 078h, 000h, 000h, 000h, 0cch, 0feh, 0feh, 0d6h, 0c6h, 000h
176	00011009	00CCFEFED6C600	<1>		
177	00011010	0000F8CCCCCCCC0000-	<1>	db	000h, 000h, 0f8h, 0cch, 0cch, 0cch, 0cch, 000h, 000h, 000h, 078h, 0cch, 0cch, 0cch, 078h, 000h
177	00011019	0078CCCCC7800	<1>		
178	00011020	0000DC66667C60F000-	<1>	db	000h, 000h, 0dch, 066h, 066h, 07ch, 060h, 0f0h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 01eh
178	00011029	0076CCCC7C0C1E	<1>		
179	00011030	0000DC766660F00000-	<1>	db	000h, 000h, 0dch, 076h, 066h, 060h, 0f0h, 000h, 000h, 000h, 07ch, 0c0h, 078h, 00ch, 0f8h, 000h
179	00011039	007CC0780CF800	<1>		
180	00011040	10307C303034180000-	<1>	db	010h, 030h, 07ch, 030h, 030h, 034h, 018h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 076h, 000h
180	00011049	00CCCCCCC7600	<1>		
181	00011050	0000CCCCC78300000-	<1>	db	000h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 000h, 000h, 000h, 0c6h, 0d6h, 0feh, 0feh, 06ch, 000h
181	00011059	00C6D6FEFE6C00	<1>		
182	00011060	0000C66C386CC60000-	<1>	db	000h, 000h, 0c6h, 06ch, 038h, 06ch, 0c6h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 07ch, 00ch, 0f8h
182	00011069	00CCCCC7C0CF8	<1>		
183	00011070	0000FC983064FC001C-	<1>	db	000h, 000h, 0fch, 098h, 030h, 064h, 0fch, 000h, 01ch, 030h, 030h, 0e0h, 030h, 030h, 01ch, 000h
183	00011079	3030E030301C00	<1>		
184	00011080	1818180018181800E0-	<1>	db	018h, 018h, 018h, 000h, 018h, 018h, 018h, 000h, 0e0h, 030h, 030h, 01ch, 030h, 030h, 0e0h, 000h
184	00011089	30301C3030E000	<1>		
185	00011090	76DC00000000000000-	<1>	db	076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 010h,

185 00011099 10386CC6C6FE00 <1>
186 000110A0 78CCC0CC78180C7800- <1>
186 000110A9 CC00CCCCC7E00 <1>
187 000110B0 1C0078CCFCC078007E- <1>
187 000110B9 C33C063E663F00 <1>
188 000110C0 CC00780C7CCC7E00E0- <1>
188 000110C9 00780C7CCC7E00 <1>
189 000110D0 3030780C7CCC7E0000- <1>
189 000110D9 0078C0C0780C38 <1>
190 000110E0 7EC33C667E603C00CC- <1>
190 000110E9 0078CCFCC07800 <1>
191 000110F0 E00078CCFCC07800CC- <1>
191 000110F9 00703030307800 <1>
192 00011100 7CC6381818183C00E0- <1>
192 00011109 00703030307800 <1>
193 00011110 C6386CC6FEC6C60030- <1>
193 00011119 300078CCFCC00 <1>
194 00011120 1C00FC607860FC0000- <1>
194 00011129 007F0C7FCC7F00 <1>
195 00011130 3E6CCCFECCCCCE0078- <1>
195 00011139 CC0078CCCC7800 <1>
196 00011140 00CC0078CCCC780000- <1>
196 00011149 E00078CCCC7800 <1>
197 00011150 78CC00CCCCC7E0000- <1>
197 00011159 E000CCCCC7E00 <1>
198 00011160 00CC00CCCC7C0CF8C3- <1>
198 00011169 183C66663C1800 <1>
199 00011170 CC00CCCCCCC780018- <1>
199 00011179 187EC0C07E1818 <1>
200 00011180 386C64F060E6FC00CC- <1>
200 00011189 CC78FC30FC3030 <1>
201 00011190 F8CCCFAC6CFC6C70E- <1>
201 00011199 1B183C1818D870 <1>
202 000111A0 1C00780C7CCC7E0038- <1>
202 000111A9 00703030307800 <1>
203 000111B0 001C0078CCCC780000- <1>
203 000111B9 1C00CCCCC7E00 <1>
204 000111C0 00F800F8CCCCC00FC- <1>
204 000111C9 00CCECFDCCC00 <1>
205 000111D0 3C6C6C3E007E000038- <1>
205 000111D9 6C6C38007C0000 <1>
206 000111E0 30003060C0CC780000- <1>
206 000111E9 0000FCC0C00000 <1>
207 000111F0 000000FC0C0C0000C3- <1>
207 000111F9 C6CCDE3366CC0F <1>
208 00011200 C3C6CCDB376FCF0318- <1>
208 00011209 18001818181800 <1>
209 00011210 003366CC6633000000- <1>
209 00011219 CC663366CC0000 <1>
210 00011220 228822882288228855- <1>
210 00011229 AA55AA55AA55AA <1>
211 00011230 DB77DBEEDB77DBEE18- <1>
211 00011239 18181818181818 <1>
212 00011240 18181818F818181818- <1>
212 00011249 18F818F818181818 <1>
213 00011250 36363636F636363600- <1>
213 00011259 000000FE363636 <1>
214 00011260 0000F818F818181836- <1>
214 00011269 36F606F6363636 <1>
215 00011270 363636363636363600- <1>
215 00011279 00FE06F6363636 <1>
216 00011280 3636F606FE00000036- <1>
216 00011289 363636FE000000 <1>
217 00011290 1818F818F800000000- <1>
217 00011299 000000F8181818 <1>
218 000112A0 181818181F00000018- <1>
218 000112A9 181818FF000000 <1>
219 000112B0 00000000FF18181818- <1>
219 000112B9 1818181F181818 <1>
220 000112C0 00000000FF00000018- <1>
220 000112C9 181818FF181818 <1>
221 000112D0 18181F181F18181836- <1>
221 000112D9 36363637363636 <1>
222 000112E0 363637303F00000000- <1>
222 000112E9 003F3037363636 <1>
223 000112F0 3636F700FF00000000- <1>
223 000112F9 00FF00F7363636 <1>
224 00011300 363637303736363600- <1>
224 00011309 00FF00FF000000 <1>
225 00011310 3636F700F736363618- <1>
225 00011319 18FF00FF000000 <1>
226 00011320 36363636FF00000000- <1>
226 00011329 00FF00FF181818 <1>
227 00011330 00000000FF36363636- <1>
227 00011339 3636363F000000 <1>
228 00011340 18181F181F00000000- <1>
228 00011349 001F181F181818 <1>
229 00011350 000000003F36363636- <1>
229 00011359 363636FF363636 <1>
230 00011360 1818FF18FF18181818- <1>
230 00011369 181818F8000000 <1>
231 00011370 000000001F181818FF- <1>
231 00011379 FFFFFFFF <1>
232 00011380 00000000FFFFFFFFF0- <1>
232 00011389 F0F0F0F0F0F0F0 <1>
233 00011390 0F0F0F0F0F0F0FFF- <1>
233 00011399 FFFFFFF00000000 <1>
234 000113A0 000076DCC8DC760000- <1>
234 000113A9 78CCF8CCF8C0C0 <1>
235 000113B0 00FCCCC0C0C0C00000- <1>
235 000113B9 FE6C6C6C6C6C00 <1>
236 000113C0 FCCC603060CCFC0000- <1>
236 000113C9 007ED8D8D87000 <1>
237 000113D0 00666666667C60C000- <1>
237 000113D9 76DC1818181800 <1>
238 000113E0 FC3078CCCC7830FC38- <1>
238 000113E9 6CC6FEC66C3800 <1>
239 000113F0 386CC6C66C6CEE001C- <1>
239 000113F9 30187CCCCC7800 <1>
240 00011400 00007EDBDB7E000006- <1>
240 00011409 0C7EDBDB7E60C0 <1>
241 00011410 3860C0F8C060380078- <1>
241 00011419 CCCCCCCCCC00 <1>
242 00011420 00FC00FC00FC000030- <1>
242 00011429 30FC303000FC00 <1>

db 078h, 0cch, 0c0h, 0cch, 078h, 018h, 00ch, 078h, 000h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
db 01ch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 07eh, 0c3h, 03ch, 006h, 03eh, 066h, 03fh, 000h
db 0cch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 0e0h, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h
db 030h, 030h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 000h, 000h, 078h, 0c0h, 0c0h, 078h, 00ch, 038h
db 07eh, 0c3h, 03ch, 066h, 07eh, 060h, 03ch, 000h, 0cch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
db 0e0h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 0cch, 000h, 070h, 030h, 030h, 030h, 078h, 000h
db 07ch, 0c6h, 038h, 018h, 018h, 018h, 03ch, 000h, 0e0h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
db 0c6h, 038h, 06ch, 0c6h, 0feh, 0c6h, 0c6h, 000h, 030h, 030h, 000h, 078h, 0cch, 0fch, 0cch, 000h
db 01ch, 000h, 0fch, 060h, 078h, 060h, 0fch, 000h, 000h, 000h, 07fh, 00ch, 07fh, 0cch, 07fh, 000h
db 03eh, 06ch, 0cch, 0feh, 0cch, 0cch, 0ceh, 000h, 078h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h
db 000h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 0e0h, 000h, 078h, 0cch, 0cch, 078h, 000h
db 078h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h, 000h, 0e0h, 000h, 0cch, 0cch, 0cch, 07eh, 000h
db 000h, 0cch, 000h, 0cch, 0cch, 07ch, 00ch, 0f8h, 0c3h, 018h, 03ch, 066h, 066h, 03ch, 018h, 000h
db 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 018h, 018h, 07eh, 0c0h, 0c0h, 07eh, 018h, 018h
db 038h, 06ch, 064h, 0f0h, 060h, 0e6h, 0fch, 000h, 0cch, 0cch, 078h, 0fch, 030h, 0fch, 030h, 030h
db 0f8h, 0cch, 0cch, 0fah, 0c6h, 0cfh, 0c6h, 0c7h, 00eh, 01bh, 018h, 03ch, 018h, 018h, 0d8h, 070h
db 01ch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 038h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
db 000h, 01ch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 01ch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
db 000h, 0f8h, 000h, 0f8h, 0cch, 0cch, 0cch, 000h, 0fch, 000h, 0cch, 0ech, 0fch, 0dch, 0cch, 000h
db 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h
db 030h, 000h, 030h, 060h, 0c0h, 0cch, 078h, 000h, 000h, 000h, 000h, 0fch, 0c0h, 0c0h, 000h, 000h
db 000h, 000h, 000h, 0fch, 00ch, 00ch, 000h, 000h, 0c3h, 0c6h, 0cch, 0deh, 033h, 066h, 0cch, 00fh
db 0c3h, 0c6h, 0cch, 0dbh, 037h, 06fh, 0cfh, 003h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 000h
db 000h, 033h, 066h, 0cch, 066h, 033h, 000h, 000h, 000h, 0cch, 066h, 033h, 066h, 0cch, 000h, 000h
db 022h, 088h, 022h, 088h, 022h, 088h, 022h, 088h, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah
db 0dbh, 077h, 0dbh, 0eeh, 0dbh, 077h, 0dbh, 0eeh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h
db 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h
db 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h
db 036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h
db 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h
db 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h
db 000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h
db 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h
db 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h
db 036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h
db 036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h
db 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h
db 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h
db 000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h
db 018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h
db 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h
db 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
db 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
db 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h
db 000h, 000h, 076h, 0dch, 0c8h, 0dch, 076h, 000h, 000h, 078h, 0cch, 0f8h, 0cch, 0f8h, 0c0h, 0c0h
db 000h, 0fch, 0cch, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 000h
db 0fch, 0cch, 060h, 030h, 060h, 0cch, 0fch, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 070h, 000h
db 000h, 066h, 066h, 066h, 066h, 07ch, 060h, 0c0h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 000h
db 0fch, 030h, 078h, 0cch, 0cch, 078h, 030h, 0fch, 038h, 06ch, 0c6h, 0feh, 0c6h, 06ch, 038h, 000h
db 038h, 06ch, 0c6h, 0c6h, 06ch, 06ch, 0eeh, 000h, 01ch, 030h, 018h, 07ch, 0cch, 0cch, 078h, 000h
db 000h, 000h, 07eh, 0dbh, 0dbh, 07eh, 000h, 000h, 006h, 00ch, 07eh, 0dbh, 0dbh, 07eh, 060h, 0c0h
db 038h, 060h, 0c0h, 0f8h, 0c0h, 060h, 038h, 000h, 078h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 000h
db 000h, 0fch, 000h, 0fch, 000h, 0fch, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 0fch, 000h

243	00011430	603018306000FC0018-	<1>	db	060h, 030h, 018h, 030h, 060h, 000h, 0fch, 000h, 018h, 030h, 060h, 030h, 018h, 000h, 0fch, 000h
243	00011439	3060301800FC00	<1>		
244	00011440	0E1B1B181818181818-	<1>	db	00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 070h
244	00011449	18181818D8D870	<1>		
245	00011450	303000FC0030300000-	<1>	db	030h, 030h, 000h, 0fch, 000h, 030h, 030h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h
245	00011459	76DC0076DC0000	<1>		
246	00011460	386C6C380000000000-	<1>	db	038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h
246	00011469	00001818000000	<1>		
247	00011470	00000000180000000F-	<1>	db	000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 0ech, 06ch, 03ch, 01ch
247	00011479	0C0C0CEC6C3C1C	<1>		
248	00011480	786C6C6C6C00000070-	<1>	db	078h, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 070h, 018h, 030h, 060h, 078h, 000h, 000h, 000h
248	00011489	18306078000000	<1>		
249	00011490	00003C3C3C3C000000-	<1>	db	000h, 000h, 03ch, 03ch, 03ch, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
249	00011499	00000000000000	<1>		
250			<1>	vgafont14:	
251	000114A0	000000000000000000-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
251	000114A9	00000000000000	<1>		
252	000114B0	7E81A58181BD99817E-	<1>	db	07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 07eh, 000h, 000h, 000h, 000h, 000h, 07eh, 0ffh
252	000114B9	00000000007EFF	<1>		
253	000114C0	DBFFFFC3E7FF7E0000-	<1>	db	0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 0feh, 0feh
253	000114C9	000000006CFEFE	<1>		
254	000114D0	FEFE7C381000000000-	<1>	db	0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch
254	000114D9	000010387CFE7C	<1>		
255	000114E0	381000000000000018-	<1>	db	038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h
255	000114E9	3C3CE7E7E71818	<1>		
256	000114F0	3C0000000000183C7E-	<1>	db	03ch, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h
256	000114F9	FFFF7E18183C00	<1>		
257	00011500	00000000000000183C-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h
257	00011509	3C180000000000	<1>		
258	00011510	FFFFFFFFFE7C3C3E7-	<1>	db	0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h
258	00011519	FFFFFFFFF0000	<1>		
259	00011520	00003C664242663C00-	<1>	db	000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh
259	00011529	000000FFFFFFF	<1>		
260	00011530	C399BDBD99C3FFFFFF-	<1>	db	0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 01eh, 00eh, 01ah, 032h
260	00011539	FF00001E0E1A32	<1>		
261	00011540	78CCCCC7800000000-	<1>	db	078h, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 066h, 066h, 03ch, 018h
261	00011549	003C6666663C18	<1>		
262	00011550	7E181800000000003F-	<1>	db	07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 070h, 0f0h
262	00011559	333F30303070F0	<1>		
263	00011560	E0000000000007F637F-	<1>	db	0e0h, 000h, 000h, 000h, 000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h
263	00011569	63636367E7E6C0	<1>		
264	00011570	000000001818DB3CE7-	<1>	db	000h, 000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h
264	00011579	3CDB1818000000	<1>		
265	00011580	000080C0E0F8FEF8E0-	<1>	db	000h, 000h, 080h, 0c0h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h, 000h
265	00011589	C0800000000000	<1>		
266	00011590	02060E3EFE3E0E0602-	<1>	db	002h, 006h, 00eh, 03eh, 0feh, 03eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch
266	00011599	0000000000183C	<1>		
267	000115A0	7E1818187E3C180000-	<1>	db	07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h
267	000115A9	00000066666666	<1>		
268	000115B0	66660066660000000-	<1>	db	066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh
268	000115B9	007FDBDBDB7B1B	<1>		
269	000115C0	1B1B1B000000007CC6-	<1>	db	01bh, 01bh, 01bh, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h
269	000115C9	60386CC6C66C38	<1>		
270	000115D0	0CC67C000000000000-	<1>	db	00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 000h
270	000115D9	000000FEFEFE00	<1>		
271	000115E0	00000000183C7E1818-	<1>	db	000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h
271	000115E9	187E3C187E0000	<1>		
272	000115F0	0000183C7E18181818-	<1>	db	000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
272	000115F9	18180000000000	<1>		
273	00011600	1818181818187E3C18-	<1>	db	018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
273	00011609	00000000000000	<1>		
274	00011610	180CFE0C1800000000-	<1>	db	018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 060h
274	00011619	00000000003060	<1>		
275	00011620	FE6030000000000000-	<1>	db	0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h
275	00011629	00000000C0C0C0	<1>		
276	00011630	FE0000000000000000-	<1>	db	0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 028h, 06ch, 0feh, 06ch, 028h, 000h
276	00011639	00286CFE6C2800	<1>		
277	00011640	00000000000001038-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 038h, 07ch, 07ch, 0feh, 0feh, 000h, 000h
277	00011649	387C7CFEFE0000	<1>		
278	00011650	0000000000FEFE7C7C-	<1>	db	000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 07ch, 07ch, 038h, 038h, 010h, 000h, 000h, 000h, 000h
278	00011659	38381000000000	<1>		
279	00011660	00000000000000000-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
279	00011669	00000000000000	<1>		
280	00011670	183C3C3C1818001818-	<1>	db	018h, 03ch, 03ch, 03ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 066h, 066h, 066h
280	00011679	00000000666666	<1>		
281	00011680	240000000000000000-	<1>	db	024h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch
281	00011689	0000006C6CFE6C	<1>		
282	00011690	6C6CFE6C6C00000018-	<1>	db	06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h
282	00011699	187CC6C2C07C06	<1>		
283	000116A0	86C67C181800000000-	<1>	db	086h, 0c6h, 07ch, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 066h
283	000116A9	00C2C60C183066	<1>		
284	000116B0	C60000000000386C6C-	<1>	db	0c6h, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 076h, 000h
284	000116B9	3876DCCCCC7600	<1>		
285	000116C0	000000303030600000-	<1>	db	000h, 000h, 000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
285	000116C9	00000000000000	<1>		
286	000116D0	00000C183030303030-	<1>	db	000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h, 000h
286	000116D9	180C0000000000	<1>		
287	000116E0	30180C0C0C0C0C1830-	<1>	db	030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
287	000116E9	00000000000000	<1>		
288	000116F0	663CFF3C6600000000-	<1>	db	066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h
288	000116F9	00000000001818	<1>		
289	00011700	7E1818000000000000-	<1>	db	07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
289	00011709	00000000000000	<1>		
290	00011710	181818300000000000-	<1>	db	018h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h
290	00011719	000000FE000000	<1>		
291	00011720	000000000000000000-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h
291	00011729	00000000181800	<1>		
292	00011730	0000000002060C1830-	<1>	db	000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h
292	00011739	60C08000000000	<1>		
293	00011740	00007CC6CEDEF6E6C6-	<1>	db	000h, 000h, 07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
293	00011749	C67C0000000000	<1>		
294	00011750	18387818181818187E-	<1>	db	018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h
294	00011759	00000000007CC6	<1>		
295	00011760	060C183060C6FE0000-	<1>	db	006h, 00ch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 006h, 006h
295	00011769	0000007CC60606	<1>		
296	00011770	3C0606C67C00000000-	<1>	db	03ch, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh
296	00011779	000C1C3C6CCCFE	<1>		
297	00011780	0C0C1E0000000000FE-	<1>	db	00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 0c6h
297	00011789	C0C0C0FC0606C6	<1>		
298	00011790	7C00000000003860C0-	<1>	db	07ch, 000h, 000h, 000h, 000h, 000h, 038h, 060h, 0c0h, 0c0h, 0fch, 0c6h, 0c6h, 0c6h, 07ch, 000h
298	00011799	C0FCC6C6C67C00	<1>		
299	000117A0	00000000FEC6060C18-	<1>	db	000h, 000h, 000h, 000h, 0feh, 0c6h, 006h, 00ch, 018h, 030h, 030h, 030h, 030h, 000h, 000h, 000h
299	000117A9	30303030000000	<1>		
300	000117B0	00007CC6C6C67CC6C6-	<1>	db	000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0

301 000117C0 7CC6C6C67E06060C78- <1>
301 000117C9 0000000000000018 <1>
302 000117D0 180000001818000000- <1>
302 000117D9 00000000181800 <1>
303 000117E0 000018183000000000- <1>
303 000117E9 00060C18306030 <1>
304 000117F0 180C06000000000000- <1>
304 000117F9 00007E00007E00 <1>
305 00011800 0000000000000603018- <1>
305 00011809 0C060C18306000 <1>
306 00011810 000000007CC6C60C18- <1>
306 00011819 18001818000000 <1>
307 00011820 00007CC6C6DEDEDED- <1>
307 00011829 C07C000000000000 <1>
308 00011830 10386CC6C6FEC6C6C6- <1>
308 00011839 0000000000FC66 <1>
309 00011840 66667C666666FC0000- <1>
309 00011849 0000003C66C2C0 <1>
310 00011850 C0C0C2663C00000000- <1>
310 00011859 00F86C66666666 <1>
311 00011860 666CF80000000000FE- <1>
311 00011869 66626878686266 <1>
312 00011870 FE0000000000FE6662- <1>
312 00011879 6878686060F000 <1>
313 00011880 000000003C66C2C0C0- <1>
313 00011889 DEC6663A000000 <1>
314 00011890 0000C6C6C6C6FEC6C6- <1>
314 00011899 C6C60000000000 <1>
315 000118A0 3C1818181818183C- <1>
315 000118A9 00000000001E0C <1>
316 000118B0 0C0C0C0CCCC780000- <1>
316 000118B9 000000E6666C6C <1>
317 000118C0 786C6C66E600000000- <1>
317 000118C9 00F06060606060 <1>
318 000118D0 6266FE0000000000C6- <1>
318 000118D9 EEFEFED6C6C6C6 <1>
319 000118E0 C60000000000C6E6F6- <1>
319 000118E9 FEDECEC6C6C600 <1>
320 000118F0 00000000386CC6C6C6- <1>
320 000118F9 C6C66C38000000 <1>
321 00011900 0000FC6666667C6060- <1>
321 00011909 60F00000000000 <1>
322 00011910 7CC6C6C6C6D6DE7C0C- <1>
322 00011919 0E00000000FC66 <1>
323 00011920 66667C6C6666E60000- <1>
323 00011929 0000007CC6C660 <1>
324 00011930 380CC6C67C00000000- <1>
324 00011939 007E7E5A181818 <1>
325 00011940 18183C0000000000C6- <1>
325 00011949 C6C6C6C6C6C6C6 <1>
326 00011950 7C0000000000C6C6C6- <1>
326 00011959 C6C6C66C381000 <1>
327 00011960 00000000C6C6C6D6- <1>
327 00011969 D6FE7C6C000000 <1>
328 00011970 0000C6C66C3838386C- <1>
328 00011979 C6C60000000000 <1>
329 00011980 666666663C1818183C- <1>
329 00011989 0000000000FEC6 <1>
330 00011990 8C183060C2C6FE0000- <1>
330 00011999 0000003C303030 <1>
331 000119A0 303030303C00000000- <1>
331 000119A9 0080C0E070381C <1>
332 000119B0 0E060200000000003C- <1>
332 000119B9 0C0C0C0C0C0C0C <1>
333 000119C0 3C00000010386CC600- <1>
333 000119C9 00000000000000 <1>
334 000119D0 0000000000000000- <1>
334 000119D9 0000000000FF00 <1>
335 000119E0 303018000000000000- <1>
335 000119E9 00000000000000 <1>
336 000119F0 000000780C7CCCCC76- <1>
336 000119F9 0000000000E060 <1>
337 00011A00 60786C6666667C0000- <1>
337 00011A09 00000000000007C <1>
338 00011A10 C6C0C0C67C00000000- <1>
338 00011A19 001C0C0C3C6CCC <1>
339 00011A20 CCCC76000000000000- <1>
339 00011A29 00007CC6FEC0C6 <1>
340 00011A30 7C0000000000386C64- <1>
340 00011A39 60F0606060F000 <1>
341 00011A40 0000000000000076CC- <1>
341 00011A49 CCCC7C0CCC7800 <1>
342 00011A50 0000E060606C766666- <1>
342 00011A59 66E60000000000 <1>
343 00011A60 18180038181818183C- <1>
343 00011A69 00000000000606 <1>
344 00011A70 000E0606060666663C- <1>
344 00011A79 000000E0606066 <1>
345 00011A80 6C786C66E600000000- <1>
345 00011A89 00381818181818 <1>
346 00011A90 18183C000000000000- <1>
346 00011A99 0000ECFED6D6D6 <1>
347 00011AA0 C60000000000000000- <1>
347 00011AA9 DC666666666600 <1>
348 00011AB0 000000000000007CC6- <1>
348 00011AB9 C6C6C67C000000 <1>
349 00011AC0 0000000000DC666666- <1>
349 00011AC9 7C6060F0000000 <1>
350 00011AD0 00000076CCCCC7C0C- <1>
350 00011AD9 0C1E0000000000 <1>
351 00011AE0 00DC76666060F00000- <1>
351 00011AE9 0000000000007C <1>
352 00011AF0 C6701CC67C00000000- <1>
352 00011AF9 00103030FC3030 <1>
353 00011B00 30361C000000000000- <1>
353 00011B09 0000CCCCCCCC <1>
354 00011B10 760000000000000000- <1>
354 00011B19 666666663C1800 <1>
355 00011B20 00000000000000C6C6- <1>
355 00011B29 D6D6FE6C000000 <1>
356 00011B30 0000000000C66C3838- <1>
356 00011B39 6CC60000000000 <1>
357 00011B40 000000C6C6C6C67E06- <1>
357 00011B49 0CF80000000000 <1>
358 00011B50 00FEC183066FE0000- <1>

db 07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h, 000h, 000h, 018h
db 018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h
db 000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h
db 018h, 00ch, 006h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h
db 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h
db 000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h, 000h
db 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h, 0fch, 066h
db 066h, 066h, 07ch, 066h, 066h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h
db 0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 0f8h, 06ch, 066h, 066h, 066h, 066h
db 066h, 06ch, 0f8h, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 062h, 066h
db 0feh, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 0f0h, 000h
db 000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 066h, 03ah, 000h, 000h, 000h
db 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h
db 03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 01eh, 00ch
db 00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 0e6h, 066h, 06ch, 06ch
db 078h, 06ch, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h
db 062h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 0c6h
db 0c6h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h
db 000h, 000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h
db 000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h
db 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h, 000h, 000h, 0fch, 066h
db 066h, 066h, 07ch, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 060h
db 038h, 00ch, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 07eh, 07eh, 05ah, 018h, 018h, 018h
db 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h
db 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 010h, 000h
db 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 07ch, 06ch, 000h, 000h, 000h
db 000h, 000h, 0c6h, 0c6h, 06ch, 038h, 038h, 038h, 06ch, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h
db 066h, 066h, 066h, 066h, 03ch, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h
db 08ch, 018h, 030h, 060h, 0c2h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 03ch, 030h, 030h, 030h
db 030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch
db 00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch
db 03ch, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h
db 030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0e0h, 060h
db 060h, 078h, 06ch, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch
db 0c6h, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 01ch, 00ch, 00ch, 03ch, 06ch, 0cch
db 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h
db 07ch, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 0f0h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 0cch, 078h, 000h
db 000h, 000h, 0e0h, 060h, 060h, 06ch, 076h, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h
db 018h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 006h, 006h
db 000h, 00eh, 006h, 006h, 006h, 006h, 066h, 066h, 03ch, 000h, 000h, 000h, 0e0h, 060h, 060h, 066h
db 06ch, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h
db 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ech, 0feh, 0d6h, 0d6h, 0d6h
db 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 000h, 000h
db 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h
db 000h, 0dch, 076h, 066h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch
db 0c6h, 070h, 01ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h
db 030h, 036h, 01ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch
db 076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 03ch, 018h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 06ch, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h, 000h, 000h, 000h, 000h
db 000h, 0feh, 0cch, 018h, 030h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 00eh, 018h, 018h, 018h

358 00011B59 00000000E181818 <1>
359 00011B60 701818180E00000000- <1>
359 00011B69 001818181800018 <1>
360 00011B70 181818000000000070- <1>
360 00011B79 1818180E181818 <1>
361 00011B80 700000000000076DC00- <1>
361 00011B89 0000000000000000 <1>
362 00011B90 00000000000010386C- <1>
362 00011B99 C6C6FE00000000 <1>
363 00011BA0 00003C66C2C0C0C266- <1>
363 00011BA9 3C0C067C000000 <1>
364 00011BB0 CCCC00CCCCCCCCC76- <1>
364 00011BB9 0000000000C1830 <1>
365 00011BC0 007CC6FEC0C67C0000- <1>
365 00011BC9 000010386C0078 <1>
366 00011BD0 0C7CCCCC7600000000- <1>
366 00011BD9 00CCCC00780C7C <1>
367 00011BE0 CCCC76000000006030- <1>
367 00011BE9 1800780C7CCCC <1>
368 00011BF0 7600000000386C3800- <1>
368 00011BF9 780C7CCCCC7600 <1>
369 00011C00 0000000000003C6660- <1>
369 00011C09 663C0C063C0000 <1>
370 00011C10 0010386C007CC6FEC0- <1>
370 00011C19 C67C0000000000 <1>
371 00011C20 CCCC007CC6FEC0C67C- <1>
371 00011C29 00000000603018 <1>
372 00011C30 007CC6FEC0C67C0000- <1>
372 00011C39 00000066660038 <1>
373 00011C40 181818183C00000000- <1>
373 00011C49 183C6600381818 <1>
374 00011C50 18183C000000006030- <1>
374 00011C59 18003818181818 <1>
375 00011C60 3C00000000C6C61038- <1>
375 00011C69 6CC6C6FEC6C600 <1>
376 00011C70 0000386C3800386CC6- <1>
376 00011C79 C6FEC6C6000000 <1>
377 00011C80 18306000FE66607C60- <1>
377 00011C89 66FE0000000000 <1>
378 00011C90 0000CC76367ED8D86E- <1>
378 00011C99 00000000003E6C <1>
379 00011CA0 CCCCFFCCCCCCE0000- <1>
379 00011CA9 000010386C007C <1>
380 00011CB0 C6C6C6C67C00000000- <1>
380 00011CB9 00C6C6007CC6C6 <1>
381 00011CC0 C6C67C000000006030- <1>
381 00011CC9 18007CC6C6C6C6 <1>
382 00011CD0 7C0000000003078CC00- <1>
382 00011CD9 CCCCCCCCCC7600 <1>
383 00011CE0 00000060301800CCCC- <1>
383 00011CE9 CCCCCC76000000 <1>
384 00011CF0 0000C6C600C6C6C6C6- <1>
384 00011CF9 7E060C780000C6 <1>
385 00011D00 C6386CC6C6C6C66C38- <1>
385 00011D09 00000000C6C600 <1>
386 00011D10 C6C6C6C6C6C67C0000- <1>
386 00011D19 000018183C6660 <1>
387 00011D20 60663C181800000000- <1>
387 00011D29 386C6460F06060 <1>
388 00011D30 60E6FC000000000066- <1>
388 00011D39 663C187E187E18 <1>
389 00011D40 18000000000F8CCCCF8- <1>
389 00011D49 C4CCDECCCCC600 <1>
390 00011D50 00000000E1B1818187E- <1>
390 00011D59 18181818D87000 <1>
391 00011D60 0018306000780C7CCC- <1>
391 00011D69 CC760000000000C <1>
392 00011D70 18300038181818183C- <1>
392 00011D79 00000000183060 <1>
393 00011D80 007CC6C6C6C67C0000- <1>
393 00011D89 000018306000CC <1>
394 00011D90 CCCCCCCC7600000000- <1>
394 00011D99 0076DC00DC6666 <1>
395 00011DA0 666666000000076DC00- <1>
395 00011DA9 C6E6F6FEDECEC6 <1>
396 00011DB0 C6000000003C6C6C3E- <1>
396 00011DB9 007E0000000000 <1>
397 00011DC0 000000386C6C38007C- <1>
397 00011DC9 00000000000000 <1>
398 00011DD0 0000303000303060C6- <1>
398 00011DD9 C67C0000000000 <1>
399 00011DE0 00000000FEC0C0C000- <1>
399 00011DE9 00000000000000 <1>
400 00011DF0 0000FE060606000000- <1>
400 00011DF9 0000C0C0C6CCD8 <1>
401 00011E00 3060DC860C183E0000- <1>
401 00011E09 C0C0C6CCD83066 <1>
402 00011E10 CE9E3E0606000000018- <1>
402 00011E19 180018183C3C3C <1>
403 00011E20 180000000000000036- <1>
403 00011E29 6CD86C36000000 <1>
404 00011E30 000000000000D86C36- <1>
404 00011E39 6CD80000000000 <1>
405 00011E40 114411441144114411- <1>
405 00011E49 441144114455AA <1>
406 00011E50 55AA55AA55AA55AA55- <1>
406 00011E59 AA55AAD77DD77 <1>
407 00011E60 DD77DD77DD77DD77DD- <1>
407 00011E69 77181818181818 <1>
408 00011E70 181818181818181818- <1>
408 00011E79 181818181818F8 <1>
409 00011E80 181818181818181818- <1>
409 00011E89 1818F818F81818 <1>
410 00011E90 181818183636363636- <1>
410 00011E99 3636F636363636 <1>
411 00011EA0 363600000000000000- <1>
411 00011EA9 FE363636363636 <1>
412 00011EB0 0000000000F818F818- <1>
412 00011EB9 18181818183636 <1>
413 00011EC0 363636F606F6363636- <1>
413 00011EC9 36363636363636 <1>
414 00011ED0 363636363636363636- <1>
414 00011ED9 360000000000FE <1>
415 00011EE0 06F636363636363636- <1>
415 00011EE9 36363636F606FE <1>

db 070h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h
db 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h
db 070h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h
db 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h, 000h
db 0cch, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 00ch, 018h, 030h
db 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 078h
db 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 000h, 078h, 00ch, 07ch
db 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch
db 076h, 000h, 000h, 000h, 000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h
db 000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
db 0cch, 0cch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h
db 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 000h, 038h
db 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h
db 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h
db 03ch, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h
db 000h, 000h, 038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h, 000h, 000h
db 018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 0cch, 076h, 036h, 07eh, 0d8h, 0d8h, 06eh, 000h, 000h, 000h, 000h, 000h, 03eh, 06ch
db 0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 07ch
db 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h, 07ch, 0c6h, 0c6h
db 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h
db 07ch, 000h, 000h, 000h, 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h
db 000h, 000h, 000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h
db 000h, 000h, 0c6h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h, 000h, 0c6h
db 0c6h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h
db 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 018h, 03ch, 066h, 060h
db 060h, 066h, 03ch, 018h, 018h, 000h, 000h, 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h
db 060h, 0e6h, 0fch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 03ch, 018h, 07eh, 018h, 07eh, 018h
db 018h, 000h, 000h, 000h, 000h, 0f8h, 0cch, 0cch, 0f8h, 0c4h, 0cch, 0deh, 0cch, 0cch, 0c6h, 000h
db 000h, 000h, 000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h
db 000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 00ch
db 018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 030h, 060h
db 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 030h, 060h, 000h, 0cch
db 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h
db 066h, 066h, 066h, 000h, 000h, 000h, 076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h
db 0c6h, 000h, 000h, 000h, 000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 0feh, 006h, 006h, 006h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c6h, 0cch, 0d8h
db 030h, 060h, 0dch, 086h, 00ch, 018h, 03eh, 000h, 000h, 0c0h, 0c0h, 0c6h, 0cch, 0d8h, 030h, 066h
db 0ceh, 09eh, 03eh, 006h, 006h, 000h, 000h, 000h, 018h, 018h, 000h, 018h, 018h, 03ch, 03ch, 03ch
db 018h, 000h, 000h, 000h, 000h, 000h, 000h, 036h, 06ch, 0d8h, 06ch, 036h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 0d8h, 06ch, 036h, 06ch, 0d8h, 000h, 000h, 000h, 000h
db 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 055h, 0aah
db 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 0ddh, 077h, 0ddh, 077h
db 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 018h, 018h, 018h, 018h, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h
db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h
db 018h, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h
db 036h, 036h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h
db 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 036h, 036h
db 036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 000h, 0feh
db 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh

416 00011EF0 0000000000000363636- <1>
416 00011EF9 36363636FE0000 <1>
417 00011F00 000000001818181818- <1>
417 00011F09 F818F80000000000 <1>
418 00011F10 000000000000000000- <1>
418 00011F19 F818181818181818 <1>
419 00011F20 181818181818181F00- <1>
419 00011F29 0000000000001818 <1>
420 00011F30 1818181818FF000000- <1>
420 00011F39 0000000000000000 <1>
421 00011F40 000000FF1818181818- <1>
421 00011F49 18181818181818 <1>
422 00011F50 181F18181818181800- <1>
422 00011F59 00000000000000FF <1>
423 00011F60 000000000000181818- <1>
423 00011F69 18181818FF1818 <1>
424 00011F70 181818181818181818- <1>
424 00011F79 1F181F1818181818 <1>
425 00011F80 181836363636363636- <1>
425 00011F89 3736363636363636 <1>
426 00011F90 363636363637303F00- <1>
426 00011F99 0000000000000000 <1>
427 00011FA0 0000003F303736363636- <1>
427 00011FA9 3636363636363636 <1>
428 00011FB0 36F700FF000000000000- <1>
428 00011FB9 00000000000000FF <1>
429 00011FC0 00F73636363636363636- <1>
429 00011FC9 36363636373037 <1>
430 00011FD0 363636363636000000- <1>
430 00011FD9 0000FF00FF0000 <1>
431 00011FE0 000000003636363636- <1>
431 00011FE9 F700F736363636 <1>
432 00011FF0 3636181818181818FF00- <1>
432 00011FF9 FF00000000000000 <1>
433 00012000 36363636363636FF00- <1>
433 00012009 0000000000000000 <1>
434 00012010 000000FF00FF18181818- <1>
434 00012019 1818180000000000 <1>
435 00012020 000000FF363636363636- <1>
435 00012029 3636363636363636 <1>
436 00012030 363F000000000000018- <1>
436 00012039 181818181F181F <1>
437 00012040 000000000000000000- <1>
437 00012049 00001F181F181818 <1>
438 00012050 181818180000000000- <1>
438 00012059 00003F3636363636 <1>
439 00012060 36363636363636363636- <1>
439 00012069 FF36363636363636 <1>
440 00012070 1818181818FF18FF18- <1>
440 00012079 1818181818181818 <1>
441 00012080 1818181818F80000000- <1>
441 00012089 0000000000000000 <1>
442 00012090 0000001F1818181818- <1>
442 00012099 18FFFFFFFFFFFFFF <1>
443 000120A0 FFFFFFFFFFFFFFFFFF00- <1>
443 000120A9 00000000000000FF <1>
444 000120B0 FFFFFFFFFFFFFFF0F0F0- <1>
444 000120B9 F0F0F0F0F0F0F0F0 <1>
445 000120C0 F0F0F0F0F0F0F0F0F0F- <1>
445 000120C9 0F0F0F0F0F0F0F0F <1>
446 000120D0 0FOFFFFFFFFFFFFFFFFF- <1>
446 000120D9 0000000000000000 <1>
447 000120E0 000000000076DCD8D8- <1>
447 000120E9 DC76000000000000 <1>
448 000120F0 00007CC6FCC6C6FCC0- <1>
448 000120F9 C0400000000FEC6 <1>
449 00012100 C6C0C0C0C0C0C0C00000- <1>
449 00012109 00000000000FE6C <1>
450 00012110 6C6C6C6C6C6C00000000- <1>
450 00012119 00FEC660301830 <1>
451 00012120 60C6FE00000000000000- <1>
451 00012129 00007ED8D8D8D8D8 <1>
452 00012130 7000000000000000066- <1>
452 00012139 6666667C6060C0 <1>
453 00012140 00000000000076DC18- <1>
453 00012149 1818181800000000 <1>
454 00012150 00007E183C6666663C- <1>
454 00012159 187E000000000000 <1>
455 00012160 386CC6C6FEC6C66C38- <1>
455 00012169 00000000000386C <1>
456 00012170 C6C6C6C6C6C6CEE0000- <1>
456 00012179 0000001E30180C <1>
457 00012180 3E6666663C000000000- <1>
457 00012189 000000007EDBDB <1>
458 00012190 7E00000000000000003- <1>
458 00012199 067EDBDBF37E60 <1>
459 000121A0 C000000000001C3060- <1>
459 000121A9 607C6060301C00 <1>
460 000121B0 00000000007CC6C6C6- <1>
460 000121B9 C6C6C6C6C6000000 <1>
461 000121C0 000000FE00000FE0000- <1>
461 000121C9 FE00000000000000 <1>
462 000121D0 0018187E18180000FF- <1>
462 000121D9 000000000003018 <1>
463 000121E0 0C060C1830007E0000- <1>
463 000121E9 0000000C183060 <1>
464 000121F0 30180C007E00000000- <1>
464 000121F9 000E1B1B181818 <1>
465 00012200 181818181818181818- <1>
465 00012209 1818181818D8D8 <1>
466 00012210 7000000000000001818- <1>
466 00012219 007E0018180000 <1>
467 00012220 00000000000076DC00- <1>
467 00012229 76DC000000000000 <1>
468 00012230 00386C6C38000000000- <1>
468 00012239 0000000000000000 <1>
469 00012240 000000001818000000- <1>
469 00012249 0000000000000000 <1>
470 00012250 000000180000000000- <1>
470 00012259 00000F0C0C0C0C <1>
471 00012260 0CEC6C3C1C00000000- <1>
471 00012269 D86C6C6C6C6C00 <1>
472 00012270 0000000000000070D8- <1>
472 00012279 3060C8F800000000 <1>
473 00012280 00000000000000007C- <1>

db 000h, 000h, 000h, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h
db 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 0ffh
db 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h
db 018h, 018h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
db 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h
db 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h
db 000h, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h
db 036h, 036h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h
db 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h
db 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 036h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h
db 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h
db 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
db 0f0h, 0f0h, 0f0h, 0f0h, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh
db 00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h
db 000h, 000h, 07ch, 0c6h, 0fch, 0c6h, 0c6h, 0fch, 0c0h, 0c0h, 040h, 000h, 000h, 000h, 0feh, 0c6h
db 0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 06ch
db 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h
db 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h
db 070h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h
db 000h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 000h, 000h
db 000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h
db 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 038h, 06ch
db 0c6h, 0c6h, 0c6h, 06ch, 06ch, 06ch, 0eeh, 000h, 000h, 000h, 000h, 01eh, 030h, 018h, 00ch
db 03eh, 066h, 066h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 0dbh, 0dbh
db 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 003h, 006h, 07eh, 0dbh, 0dbh, 0f3h, 07eh, 060h
db 0c0h, 000h, 000h, 000h, 000h, 000h, 01ch, 030h, 060h, 060h, 07ch, 060h, 060h, 030h, 01ch, 000h
db 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h
db 000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 0feh, 000h, 000h, 000h, 000h, 000h
db 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 030h, 018h
db 00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 00ch, 018h, 030h, 060h
db 030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 00eh, 01bh, 01bh, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h
db 070h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h
db 000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 00ch
db 00ch, 0ech, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h

531 00012619 30303000000000 <1>
532 00012620 00007CC6C6C67CC6C6- <1>
532 00012629 C6C67C0000000000 <1>
533 00012630 00007CC6C6C67E0606- <1>
533 00012639 060C780000000000 <1>
534 00012640 000000001818000000- <1>
534 00012649 1818000000000000 <1>
535 00012650 000000001818000000- <1>
535 00012659 1818300000000000 <1>
536 00012660 000000060C18306030- <1>
536 00012669 180C060000000000 <1>
537 00012670 000000000007E00007E- <1>
537 00012679 0000000000000000 <1>
538 00012680 0000006030180C060C- <1>
538 00012689 1830600000000000 <1>
539 00012690 00007CC6C60C181818- <1>
539 00012699 0018180000000000 <1>
540 000126A0 0000007CC6C6DEDEDE- <1>
540 000126A9 DCC07C0000000000 <1>
541 000126B0 000010386CC6C6FEC6- <1>
541 000126B9 C6C6C60000000000 <1>
542 000126C0 0000FC6666667C6666- <1>
542 000126C9 6666FC0000000000 <1>
543 000126D0 00003C66C2C0C0C0C0- <1>
543 000126D9 C2663C0000000000 <1>
544 000126E0 0000F86C6666666666- <1>
544 000126E9 666CF80000000000 <1>
545 000126F0 0000FE666268786860- <1>
545 000126F9 6266FE0000000000 <1>
546 00012700 0000FE666268786860- <1>
546 00012709 6060F00000000000 <1>
547 00012710 00003C66C2C0C0DEC6- <1>
547 00012719 C6663A0000000000 <1>
548 00012720 0000C6C6C6C6FEC6C6- <1>
548 00012729 C6C6C60000000000 <1>
549 00012730 00003C181818181818- <1>
549 00012739 18183C0000000000 <1>
550 00012740 00001E0C0C0C0C0CCCC- <1>
550 00012749 CCCC780000000000 <1>
551 00012750 0000E6666666C78786C- <1>
551 00012759 6666E60000000000 <1>
552 00012760 0000F0606060606060- <1>
552 00012769 6266FE0000000000 <1>
553 00012770 0000C3E7FFFFDBC3C3- <1>
553 00012779 C3C3C30000000000 <1>
554 00012780 0000C6E6F6FEDECEC6- <1>
554 00012789 C6C6C60000000000 <1>
555 00012790 00007CC6C6C6C6C6C6- <1>
555 00012799 C6C67C0000000000 <1>
556 000127A0 0000FC6666667C6060- <1>
556 000127A9 6060F00000000000 <1>
557 000127B0 00007CC6C6C6C6C6C6- <1>
557 000127B9 D6DE7C0C0E0000 <1>
558 000127C0 0000FC6666667C6C66- <1>
558 000127C9 6666E60000000000 <1>
559 000127D0 00007CC6C660380C06- <1>
559 000127D9 C6C67C0000000000 <1>
560 000127E0 0000FFDB9918181818- <1>
560 000127E9 18183C0000000000 <1>
561 000127F0 0000C6C6C6C6C6C6- <1>
561 000127F9 C6C67C0000000000 <1>
562 00012800 0000C3C3C3C3C3C3C3- <1>
562 00012809 663C180000000000 <1>
563 00012810 0000C3C3C3C3C3DBDB- <1>
563 00012819 FF66660000000000 <1>
564 00012820 0000C3C3663C18183C- <1>
564 00012829 66C3C30000000000 <1>
565 00012830 0000C3C3C3663C1818- <1>
565 00012839 18183C0000000000 <1>
566 00012840 0000FFC3860C183060- <1>
566 00012849 C1C3FF0000000000 <1>
567 00012850 00003C303030303030- <1>
567 00012859 30303C0000000000 <1>
568 00012860 00000080C0E070381C- <1>
568 00012869 0E06020000000000 <1>
569 00012870 00003C0C0C0C0C0C0C- <1>
569 00012879 0C0C3C0000000000 <1>
570 00012880 10386CC6000000000000- <1>
570 00012889 0000000000000000 <1>
571 00012890 00000000000000000000- <1>
571 00012899 00000000FF0000 <1>
572 000128A0 303018000000000000- <1>
572 000128A9 0000000000000000 <1>
573 000128B0 0000000000780C7CCC- <1>
573 000128B9 CCCC760000000000 <1>
574 000128C0 0000E06060786C6666- <1>
574 000128C9 66667C0000000000 <1>
575 000128D0 00000000007CC6C0C0- <1>
575 000128D9 C0C67C0000000000 <1>
576 000128E0 00001C0C0C3C6CCCCC- <1>
576 000128E9 CCCC760000000000 <1>
577 000128F0 00000000007CC6FEC0- <1>
577 000128F9 C0C67C0000000000 <1>
578 00012900 0000386C6460F06060- <1>
578 00012909 6060F00000000000 <1>
579 00012910 000000000076CCCCC- <1>
579 00012919 CCCC7C0CCC7800 <1>
580 00012920 0000E060606C766666- <1>
580 00012929 6666E60000000000 <1>
581 00012930 000018180038181818- <1>
581 00012939 18183C0000000000 <1>
582 00012940 00000606000E060606- <1>
582 00012949 06060666663C00 <1>
583 00012950 0000E06060666C7878- <1>
583 00012959 6C66E60000000000 <1>
584 00012960 000038181818181818- <1>
584 00012969 18183C0000000000 <1>
585 00012970 0000000000E6FFDBDB- <1>
585 00012979 DBDBDB0000000000 <1>
586 00012980 0000000000DC666666- <1>
586 00012989 6666660000000000 <1>
587 00012990 00000000007CC6C6C6- <1>
587 00012999 C6C67C0000000000 <1>
588 000129A0 0000000000DC666666- <1>
588 000129A9 66667C6060F000 <1>

db 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
db 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 006h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h, 000h, 000h, 000h
db 000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h
db 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
db 000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 066h, 066h, 066h, 066h, 0fch, 000h, 000h, 000h, 000h
db 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h
db 000h, 000h, 0f8h, 06ch, 066h, 066h, 066h, 066h, 066h, 066h, 06ch, 0f8h, 000h, 000h, 000h, 000h
db 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 062h, 066h, 0feh, 000h, 000h, 000h, 000h
db 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
db 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 0c6h, 066h, 03ah, 000h, 000h, 000h, 000h
db 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
db 000h, 000h, 03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
db 000h, 000h, 01eh, 00ch, 00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h
db 000h, 000h, 0e6h, 066h, 066h, 06ch, 078h, 078h, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
db 000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 000h, 000h, 000h
db 000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 000h
db 000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
db 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
db 000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
db 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h
db 000h, 000h, 0fch, 066h, 066h, 066h, 07ch, 06ch, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
db 000h, 000h, 07ch, 0c6h, 0c6h, 060h, 038h, 00ch, 006h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
db 000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
db 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
db 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h
db 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 000h
db 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 000h
db 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
db 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 060h, 0c1h, 0c3h, 0ffh, 000h, 000h, 000h, 000h
db 000h, 000h, 03ch, 030h, 030h, 030h, 030h, 030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch, 00eh, 006h, 002h, 000h, 000h, 000h, 000h
db 000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 03ch, 000h, 000h, 000h, 000h
db 010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h
db 030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
db 000h, 000h, 0e0h, 060h, 060h, 078h, 06ch, 066h, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c0h, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
db 000h, 000h, 01ch, 00ch, 00ch, 03ch, 06ch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
db 000h, 000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 0cch, 0cch, 07ch, 00ch, 0cch, 078h, 000h
db 000h, 000h, 0e0h, 060h, 060h, 06ch, 076h, 066h, 066h, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h
db 000h, 000h, 018h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
db 000h, 000h, 006h, 006h, 000h, 00eh, 006h, 006h, 006h, 006h, 006h, 006h, 066h, 066h, 03ch, 000h
db 000h, 000h, 0e0h, 060h, 060h, 066h, 06ch, 078h, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h
db 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h

589	000129B0	0000000000076CCCCC-	<1>	db	000h, 000h, 000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h
589	000129B9	CCCC7C0C0C1E00	<1>		
590	000129C0	0000000000DC766660-	<1>	db	000h, 000h, 000h, 000h, 000h, 0dch, 076h, 066h, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h
590	000129C9	6060F000000000	<1>		
591	000129D0	000000000007CC66038-	<1>	db	000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h
591	000129D9	0CC67C00000000	<1>		
592	000129E0	0000103030FC303030-	<1>	db	000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h, 030h, 030h, 036h, 01ch, 000h, 000h, 000h, 000h
592	000129E9	30361C00000000	<1>		
593	000129F0	00000000000CCCCCCC-	<1>	db	000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
593	000129F9	CCCC7600000000	<1>		
594	00012A00	0000000000C3C3C3C3-	<1>	db	000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h
594	00012A09	663C1800000000	<1>		
595	00012A10	0000000000C3C3C3DB-	<1>	db	000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h
595	00012A19	DBFF6600000000	<1>		
596	00012A20	0000000000C3663C18-	<1>	db	000h, 000h, 000h, 000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h
596	00012A29	3C66C300000000	<1>		
597	00012A30	0000000000C6C6C6C6-	<1>	db	000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h
597	00012A39	C6C67E060CF800	<1>		
598	00012A40	0000000000FECC1830-	<1>	db	000h, 000h, 000h, 000h, 000h, 0feh, 0cch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h
598	00012A49	60C6FE00000000	<1>		
599	00012A50	00000E181818701818-	<1>	db	000h, 000h, 00eh, 018h, 018h, 018h, 070h, 018h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h
599	00012A59	18180E00000000	<1>		
600	00012A60	000018181818001818-	<1>	db	000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
600	00012A69	18181800000000	<1>		
601	00012A70	0000701818180E1818-	<1>	db	000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h, 018h, 070h, 000h, 000h, 000h, 000h
601	00012A79	18187000000000	<1>		
602	00012A80	000076DC0000000000-	<1>	db	000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
602	00012A89	00000000000000	<1>		
603	00012A90	0000000010386CC6C6-	<1>	db	000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h
603	00012A99	C6FE0000000000	<1>		
604	00012AA0	00003C66C2C0C0C0C2-	<1>	db	000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h
604	00012AA9	663C0C067C0000	<1>		
605	00012AB0	0000CC00000CCCCCCC-	<1>	db	000h, 000h, 0cch, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
605	00012AB9	CCCC7600000000	<1>		
606	00012AC0	000C1830007CC6FEC0-	<1>	db	000h, 00ch, 018h, 030h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
606	00012AC9	C0C67C00000000	<1>		
607	00012AD0	0010386C00780C7CCC-	<1>	db	000h, 010h, 038h, 06ch, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
607	00012AD9	CCCC7600000000	<1>		
608	00012AE0	0000CC0000780C7CCC-	<1>	db	000h, 000h, 0cch, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
608	00012AE9	CCCC7600000000	<1>		
609	00012AF0	0060301800780C7CCC-	<1>	db	000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
609	00012AF9	CCCC7600000000	<1>		
610	00012B00	00386C3800780C7CCC-	<1>	db	000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
610	00012B09	CCCC7600000000	<1>		
611	00012B10	000000003C66606066-	<1>	db	000h, 000h, 000h, 000h, 03ch, 066h, 060h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h, 000h
611	00012B19	3C0C063C000000	<1>		
612	00012B20	0010386C007CC6FEC0-	<1>	db	000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
612	00012B29	C0C67C00000000	<1>		
613	00012B30	0000C600007CC6FEC0-	<1>	db	000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
613	00012B39	C0C67C00000000	<1>		
614	00012B40	00603018007CC6FEC0-	<1>	db	000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
614	00012B49	C0C67C00000000	<1>		
615	00012B50	000066000038181818-	<1>	db	000h, 000h, 066h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
615	00012B59	18183C00000000	<1>		
616	00012B60	00183C660038181818-	<1>	db	000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
616	00012B69	18183C00000000	<1>		
617	00012B70	006030180038181818-	<1>	db	000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
617	00012B79	18183C00000000	<1>		
618	00012B80	00C60010386CC6C6FE-	<1>	db	000h, 0c6h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
618	00012B89	C6C6C600000000	<1>		
619	00012B90	386C3800386CC6C6FE-	<1>	db	038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
619	00012B99	C6C6C600000000	<1>		
620	00012BA0	18306000FE66607C60-	<1>	db	018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 060h, 066h, 0feh, 000h, 000h, 000h, 000h
620	00012BA9	6066FE00000000	<1>		
621	00012BB0	00000000006E3B1B7E-	<1>	db	000h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 000h
621	00012BB9	D8DC7700000000	<1>		
622	00012BC0	00003E6CCCCCFECCCC-	<1>	db	000h, 000h, 03eh, 06ch, 0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h
622	00012BC9	CCCCCE00000000	<1>		
623	00012BD0	0010386C007CC6C6C6-	<1>	db	000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
623	00012BD9	C6C67C00000000	<1>		
624	00012BE0	0000C600007CC6C6C6-	<1>	db	000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
624	00012BE9	C6C67C00000000	<1>		
625	00012BF0	00603018007CC6C6C6-	<1>	db	000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
625	00012BF9	C6C67C00000000	<1>		
626	00012C00	003078CC00CCCCCCCC-	<1>	db	000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
626	00012C09	CCCC7600000000	<1>		
627	00012C10	0060301800CCCCCCCC-	<1>	db	000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
627	00012C19	CCCC7600000000	<1>		
628	00012C20	0000C60000C6C6C6C6-	<1>	db	000h, 000h, 0c6h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h
628	00012C29	C6C67E060C7800	<1>		
629	00012C30	00C6007CC6C6C6C6C6-	<1>	db	000h, 0c6h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
629	00012C39	C6C67C00000000	<1>		
630	00012C40	00C600C6C6C6C6C6C6-	<1>	db	000h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
630	00012C49	C6C67C00000000	<1>		
631	00012C50	0018187EC3C0C0C0C3-	<1>	db	000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h
631	00012C59	7E181800000000	<1>		
632	00012C60	00386C6460F0606060-	<1>	db	000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0e6h, 0fch, 000h, 000h, 000h, 000h
632	00012C69	60E6FC00000000	<1>		
633	00012C70	0000C3663C18FF18FF-	<1>	db	000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h
633	00012C79	18181800000000	<1>		
634	00012C80	00FC66667C62666F66-	<1>	db	000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h, 000h
634	00012C89	6666F300000000	<1>		
635	00012C90	000E1B18187E181818-	<1>	db	000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h, 000h
635	00012C99	181818D8700000	<1>		
636	00012CA0	0018306000780C7CCC-	<1>	db	000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
636	00012CA9	CCCC7600000000	<1>		
637	00012CB0	000C18300038181818-	<1>	db	000h, 00ch, 018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
637	00012CB9	18183C00000000	<1>		
638	00012CC0	00183060007CC6C6C6-	<1>	db	000h, 018h, 030h, 060h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
638	00012CC9	C6C67C00000000	<1>		
639	00012CD0	0018306000CCCCCCCC-	<1>	db	000h, 018h, 030h, 060h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
639	00012CD9	CCCC7600000000	<1>		
640	00012CE0	000076DC00DC666666-	<1>	db	000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h
640	00012CE9	66666600000000	<1>		
641	00012CF0	76DC00C6E6F6FEDECE-	<1>	db	076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
641	00012CF9	C6C6C600000000	<1>		
642	00012D00	003C6C6C3E007E0000-	<1>	db	000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
642	00012D09	00000000000000	<1>		
643	00012D10	00386C6C38007C0000-	<1>	db	000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
643	00012D19	00000000000000	<1>		
644	00012D20	0000303000303060C0-	<1>	db	000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c0h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
644	00012D29	C6C67C00000000	<1>		
645	00012D30	000000000000FEC0C0-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h
64					

646 00012D49 0606000000000000 <1>
647 00012D50 00C0C0C2C6CC183060- <1>
647 00012D59 CE9B060C1F0000 <1>
648 00012D60 00C0C0C2C6CC183066- <1>
648 00012D69 CE963E06060000 <1>
649 00012D70 00001818001818183C- <1>
649 00012D79 3C3C180000000000 <1>
650 00012D80 0000000000366CD86C- <1>
650 00012D89 3600000000000000 <1>
651 00012D90 0000000000D86C366C- <1>
651 00012D99 D800000000000000 <1>
652 00012DA0 114411441144114411- <1>
652 00012DA9 44114411441144 <1>
653 00012DB0 55AA55AA55AA55AA55- <1>
653 00012DB9 AA55AA55AA55AA <1>
654 00012DC0 DD77DD77DD77DD77DD- <1>
654 00012DC9 77DD77DD77DD77 <1>
655 00012DD0 181818181818181818- <1>
655 00012DD9 18181818181818 <1>
656 00012DE0 18181818181818F818- <1>
656 00012DE9 18181818181818 <1>
657 00012DF0 1818181818F818F818- <1>
657 00012DF9 18181818181818 <1>
658 00012E00 36363636363636F636- <1>
658 00012E09 36363636363636 <1>
659 00012E10 00000000000000FE36- <1>
659 00012E19 36363636363636 <1>
660 00012E20 0000000000F818F818- <1>
660 00012E29 18181818181818 <1>
661 00012E30 3636363636F606F636- <1>
661 00012E39 36363636363636 <1>
662 00012E40 363636363636363636- <1>
662 00012E49 36363636363636 <1>
663 00012E50 0000000000FE06F636- <1>
663 00012E59 36363636363636 <1>
664 00012E60 3636363636F606FE00- <1>
664 00012E69 0000000000000000 <1>
665 00012E70 36363636363636FE00- <1>
665 00012E79 0000000000000000 <1>
666 00012E80 1818181818F818F800- <1>
666 00012E89 0000000000000000 <1>
667 00012E90 00000000000000F818- <1>
667 00012E99 18181818181818 <1>
668 00012EA0 181818181818181F00- <1>
668 00012EA9 0000000000000000 <1>
669 00012EB0 18181818181818FF00- <1>
669 00012EB9 0000000000000000 <1>
670 00012EC0 00000000000000FF18- <1>
670 00012EC9 18181818181818 <1>
671 00012ED0 181818181818181F18- <1>
671 00012ED9 18181818181818 <1>
672 00012EE0 00000000000000FF00- <1>
672 00012EE9 0000000000000000 <1>
673 00012EF0 18181818181818FF18- <1>
673 00012EF9 18181818181818 <1>
674 00012F00 18181818181F181F18- <1>
674 00012F09 18181818181818 <1>
675 00012F10 363636363636363736- <1>
675 00012F19 36363636363636 <1>
676 00012F20 363636363637303F00- <1>
676 00012F29 0000000000000000 <1>
677 00012F30 00000000003F303736- <1>
677 00012F39 36363636363636 <1>
678 00012F40 3636363636F700FF00- <1>
678 00012F49 0000000000000000 <1>
679 00012F50 0000000000FF00F736- <1>
679 00012F59 36363636363636 <1>
680 00012F60 363636363637303736- <1>
680 00012F69 36363636363636 <1>
681 00012F70 0000000000FF00FF00- <1>
681 00012F79 0000000000000000 <1>
682 00012F80 3636363636F700F736- <1>
682 00012F89 36363636363636 <1>
683 00012F90 1818181818FF00FF00- <1>
683 00012F99 0000000000000000 <1>
684 00012FA0 36363636363636FF00- <1>
684 00012FA9 0000000000000000 <1>
685 00012FB0 0000000000FF00FF18- <1>
685 00012FB9 18181818181818 <1>
686 00012FC0 00000000000000FF36- <1>
686 00012FC9 36363636363636 <1>
687 00012FD0 363636363636363F00- <1>
687 00012FD9 0000000000000000 <1>
688 00012FE0 18181818181F181F00- <1>
688 00012FE9 0000000000000000 <1>
689 00012FF0 00000000001F181F18- <1>
689 00012FF9 18181818181818 <1>
690 00013000 000000000000003F36- <1>
690 00013009 36363636363636 <1>
691 00013010 36363636363636FF36- <1>
691 00013019 36363636363636 <1>
692 00013020 1818181818FF18FF18- <1>
692 00013029 18181818181818 <1>
693 00013030 18181818181818F800- <1>
693 00013039 0000000000000000 <1>
694 00013040 000000000000001F18- <1>
694 00013049 18181818181818 <1>
695 00013050 FFFFFFFFFFFFFFFFFFFF- <1>
695 00013059 FFFFFFFFFFFFFFFFFFFF- <1>
696 00013060 00000000000000FFFF- <1>
696 00013069 FFFFFFFFFFFFFFFFFFFF- <1>
697 00013070 F0F0F0F0F0F0F0F0F0- <1>
697 00013079 F0F0F0F0F0F0F0F0 <1>
698 00013080 F0F0F0F0F0F0F0F0F0- <1>
698 00013089 F0F0F0F0F0F0F0F0 <1>
699 00013090 FFFFFFFFFFFFFFFFFF0000- <1>
699 00013099 0000000000000000 <1>
700 000130A0 000000000076DCD8D8- <1>
700 000130A9 D8DC760000000000 <1>
701 000130B0 000078CCCCCD8CCC6- <1>
701 000130B9 C6C6C00000000000 <1>
702 000130C0 0000FEC6C6C0C0C0C0- <1>
702 000130C9 C0C0C00000000000 <1>
703 000130D0 00000000FE6C6C6C6C- <1>
703 000130D9 6C6C6C0000000000 <1>

db 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh, 000h, 000h
db 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h, 006h, 000h, 000h
db 000h, 000h, 018h, 018h, 000h, 018h, 018h, 018h, 03ch, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 036h, 06ch, 0d8h, 06ch, 036h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 0d8h, 06ch, 036h, 06ch, 0d8h, 000h, 000h, 000h, 000h, 000h, 000h
db 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h, 011h, 044h
db 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah
db 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h
db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 000h, 000h, 000h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 000h, 000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
db 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
db 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
db 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh
db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h, 000h
db 000h, 000h, 078h, 0cch, 0cch, 0cch, 0d8h, 0cch, 0c6h, 0c6h, 0c6h, 0cch, 000h, 000h, 000h, 000h
db 000h, 000h, 0feh, 0c6h, 0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h
db 000h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h

704	000130E0	000000FEC660301830-	<1>	db	000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h									
704	000130E9	60C6FE00000000	<1>											
705	000130F0	00000000007ED8D8D8-	<1>	db	000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h									
705	000130F9	D8D87000000000	<1>											
706	00013100	000000006666666666-	<1>	db	000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h, 000h, 000h, 000h									
706	00013109	7C6060C0000000	<1>											
707	00013110	0000000076DC181818-	<1>	db	000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h									
707	00013119	18181800000000	<1>											
708	00013120	0000007E183C666666-	<1>	db	000h, 000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h									
708	00013129	3C187E00000000	<1>											
709	00013130	000000386CC6C6FEC6-	<1>	db	000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h									
709	00013139	C66C3800000000	<1>											
710	00013140	0000386CC6C6C666C6-	<1>	db	000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 06ch, 06ch, 06ch, 0eeh, 000h, 000h, 000h, 000h									
710	00013149	6C6CEE00000000	<1>											
711	00013150	00001E30180C3E6666-	<1>	db	000h, 000h, 01eh, 030h, 018h, 00ch, 03eh, 066h, 066h, 066h, 066h, 03ch, 000h, 000h, 000h, 000h									
711	00013159	66663C00000000	<1>											
712	00013160	00000000007EDBDBDB-	<1>	db	000h, 000h, 000h, 000h, 000h, 07eh, 0dbh, 0dbh, 0dbh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h									
712	00013169	7E000000000000	<1>											
713	00013170	00000003067EDBDBF3-	<1>	db	000h, 000h, 000h, 003h, 006h, 07eh, 0dbh, 0dbh, 0f3h, 07eh, 060h, 0c0h, 000h, 000h, 000h, 000h									
713	00013179	7E60C000000000	<1>											
714	00013180	00001C3060607C6060-	<1>	db	000h, 000h, 01ch, 030h, 060h, 060h, 07ch, 060h, 060h, 060h, 030h, 01ch, 000h, 000h, 000h, 000h									
714	00013189	60301C00000000	<1>											
715	00013190	0000007CC6C6C6C6C6-	<1>	db	000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h									
715	00013199	C6C6C600000000	<1>											
716	000131A0	00000000FE0000FE00-	<1>	db	000h, 000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h									
716	000131A9	00FE0000000000	<1>											
717	000131B0	0000000018187E1818-	<1>	db	000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h									
717	000131B9	0000FF00000000	<1>											
718	000131C0	00000030180C060C18-	<1>	db	000h, 000h, 000h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h									
718	000131C9	30007E00000000	<1>											
719	000131D0	0000000C1830603018-	<1>	db	000h, 000h, 000h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h									
719	000131D9	0C007E00000000	<1>											
720	000131E0	00000E1B1B18181818-	<1>	db	000h, 000h, 00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h									
720	000131E9	18181818181818	<1>											
721	000131F0	18181818181818D8-	<1>	db	018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h									
721	000131F9	D8D87000000000	<1>											
722	00013200	000000001818007E00-	<1>	db	000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h									
722	00013209	18180000000000	<1>											
723	00013210	000000000076DC0076-	<1>	db	000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h									
723	00013219	DC000000000000	<1>											
724	00013220	00386C6C3800000000-	<1>	db	000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h									
724	00013229	00000000000000	<1>											
725	00013230	0000000000000001818-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h									
725	00013239	00000000000000	<1>											
726	00013240	000000000000000018-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h									
726	00013249	00000000000000	<1>											
727	00013250	000F0C0C0C0C0CEC6C-	<1>	db	000h, 00fh, 00ch, 00ch, 00ch, 00ch, 00ch, 0ech, 06ch, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h									
727	00013259	6C3C1C00000000	<1>											
728	00013260	00D86C6C6C6C6C0000-	<1>	db	000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h									
728	00013269	00000000000000	<1>											
729	00013270	0070D83060C8F80000-	<1>	db	000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h									
729	00013279	00000000000000	<1>											
730	00013280	000000007C7C7C7C7C-	<1>	db	000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h, 000h, 000h, 000h									
730	00013289	7C7C0000000000	<1>											
731	00013290	000000000000000000-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h									
731	00013299	00000000000000	<1>											
732		<1>		vgafontl4alt:										
733	000132A0	1D000000002466FF66-	<1>	db	01dh, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 022h									
733	000132A9	24000000000002	<1>											
734	000132B0	006363632200000000-	<1>	db	000h, 063h, 063h, 063h, 022h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02bh, 000h									
734	000132B9	000000000002B00	<1>											
735	000132C0	0000181818FF181818-	<1>	db	000h, 000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 02dh, 000h, 000h									
735	000132C9	000000002D0000	<1>											
736	000132D0	00000000FF00000000-	<1>	db	000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 04dh, 000h, 000h, 0c3h									
736	000132D9	0000004D0000C3	<1>											
737	000132E0	E7FFDBC3C3C3C3C300-	<1>	db	0e7h, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh									
737	000132E9	0000540000FFDB	<1>											
738	000132F0	9918181818183C0000-	<1>	db	099h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h									
738	000132F9	00560000C3C3C3	<1>											
739	00013300	C3C3C3663C18000000-	<1>	db	0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h									
739	00013309	570000C3C3C3C3	<1>											
740	00013310	DBDBFF666600000058-	<1>	db	0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h									
740	00013319	0000C3C3663C18	<1>											
741	00013320	3C66C3C300000005900-	<1>	db	03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h									
741	00013329	00C3C3C3663C18	<1>											
742	00013330	18183C00000005A0000-	<1>	db	018h, 018h, 03ch, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 061h									
742	00013339	FFC3860C183061	<1>											
743	00013340	C3FF0000006D000000-	<1>	db	0c3h, 0ffh, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh									
743	00013349	0000E6FFDBDBDB	<1>											
744	00013350	DB0000007600000000-	<1>	db	0dbh, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h									
744	00013359	00C3C3C3663C18	<1>											
745	00013360	000000770000000000-	<1>	db	000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h									
745	00013369	C3C3DBDBFF6660	<1>											
746	00013370	000091000000006E3B-	<1>	db	000h, 000h, 091h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h									
746	00013379	1B7ED8DC770000	<1>											
747	00013380	009B0018187EC3C0C0-	<1>	db	000h, 09bh, 000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h									
747	00013389	C37E1818000000	<1>											
748	00013390	9D0000C3663C18FF18-	<1>	db	09dh, 000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 000h, 000h, 000h, 09eh									
748	00013399	FF18180000009E	<1>											
749	000133A0	00FC66667C62666F66-	<1>	db	000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 0f3h, 000h, 000h, 000h, 0f1h, 000h									
749	000133A9	66F3000000F100	<1>											
750	000133B0	00181818FF18181800-	<1>	db	000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 0ffh, 000h, 000h, 000h, 0f6h, 000h, 000h									
750	000133B9	FF000000F60000	<1>											
751	000133C0	18180000FF00001818-	<1>	db	018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h									
751	000133C9	00000000	<1>											
752		<1>		vgafontl6alt:										
753	000133CD	1D00000000002466FF-	<1>	db	01dh, 000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h									
753	000133D6	66240000000000	<1>											
754	000133DD	003000003C66C3C3DB-	<1>	db	000h, 030h, 000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h									
754	000133E6	DBC3C3663C0000	<1>											
755	000133ED	00004D0000C3E7FFFF-	<1>	db	000h, 000h, 04dh, 000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h									
755	000133F6	DBC3C3C3C3C300	<1>											
756	000133FD	000000540000FFDB99-	<1>	db	000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch									
756	00013406	1818181818183C	<1>											
757	0001340D	00000000560000C3C3-	<1>	db	000h, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch									
757	00013416	C3C3C3C3C3663C	<1>											
758	0001341D	1800000000570000C3-	<1>	db	018h, 000h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh									
758	00013426	C3C3C3C3DBDBFF	<1>											
759	0001342D	666600000000580000-	<1>	db	066h, 066h, 000h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch									
759	00013436	C3C3663C18183C	<1>											
760	0001343D	66C3C3000000005900-	<1>	db	066h, 0c3h, 0c3h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h									
760	00013446	00C3C3C3663C18	<1>											
761														

```
762 00013466 6D00000000000E6 <1>
763 0001346D FFDDBDBDBDB000000- <1>
763 00013476 0076000000000000 <1>
764 0001347D C3C3C3C3663C180000- <1>
764 00013486 0000770000000000 <1>
765 0001348D 00C3C3C3DBDBFF6600- <1>
765 00013496 0000007800000000 <1>
766 0001349D 0000C3663C183C66C3- <1>
766 000134A6 00000000910000 <1>
767 000134AD 0000006E3B1B7ED8DC- <1>
767 000134B6 770000000009B00 <1>
768 000134BD 18187EC3C0C0C0C37E- <1>
768 000134C6 18180000000009D <1>
769 000134CD 0000C3663C18FF18FF- <1>
769 000134D6 1818180000000000 <1>
770 000134DD 9E00FC66667C62666F- <1>
770 000134E6 666666F30000000 <1>
771 000134ED 00AB00C0C0C2C6CC18- <1>
771 000134F6 3060CE9B060C1F <1>
772 000134FD 0000AC00C0C0C2C6CC- <1>
772 00013506 183066CE963E06 <1>
773 0001350D 06000000 <1>
2643
2644 00013511 90
2645
2646
2647
2648
2649
2650
2651 00013512 B207
2652 00013514 0100
2653 00013516 0100
2654 00013518 0000
2655 0001351A 0000
2656 0001351C 0000
2657
2658
2659 0001351E 0000
2660 00013520 1F00
2661 00013522 3B00
2662 00013524 5A00
2663 00013526 7800
2664 00013528 9700
2665 0001352A B500
2666 0001352C D400
2667 0001352E F300
2668 00013530 1101
2669 00013532 3001
2670 00013534 4E01
2671
2672
2673
2674
2675
2676
2677
2678
2679 00013536 <res 00000002>
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689 00013538 <res 00000280>
2690
2691
2692
2693
2694
2695
2696
2697 000137B8 <res 00000002>
2698 000137BA <res 00000002>
2699
2700 000137BC <res 00000004>
2701 000137C0 <res 00000002>
2702 000137C2 <res 00000002>
2703 000137C4 <res 00000004>
2704 000137C8 <res 00000002>
2705 000137CA <res 00000002>
2706 000137CC <res 00000004>
2707 000137D0 <res 00000002>
2708 000137D2 <res 00000002>
2709
2710 000137D4 <res 00000004>
2711 000137D8 <res 00000004>
2712 000137DC <res 00000004>
2713
2714 000137E0 <res 00000004>
2715 000137E4 <res 00000004>
2716 000137E8 <res 00000004>
2717 000137EC <res 00000004>
2718 000137F0 <res 00000004>
2719 000137F4 <res 00000004>
2720 000137F8 <res 00000004>
2721 000137FC <res 00000004>
2722
2723 00013800 <res 00000002>
2724 00013802 <res 00000002>
2725 00013804 <res 00000002>
```

```
db 0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h
db 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h
db 000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h, 078h, 000h, 000h, 000h
db 000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h, 091h, 000h, 000h
db 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 000h, 09bh, 000h
db 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 09dh
db 000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h
db 09eh, 000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h
db 000h, 0abh, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh
db 000h, 000h, 0ach, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h
db 006h, 000h, 000h, 000h
```

align 2

```
; EPOCH Variables
; 13/04/2015 - Retro UNIX 386 v1 Beginning
; 09/04/2013 epoch variables
; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013
;
```

```
year:      dw 1970
month:     dw 1
day:       dw 1
hour:      dw 0
minute:    dw 0
second:    dw 0
```

```
DMonth:
        dw 0
        dw 31
        dw 59
        dw 90
        dw 120
        dw 151
        dw 181
        dw 212
        dw 243
        dw 273
        dw 304
        dw 334
```

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

bss_start:

ABSOLUTE bss_start

alignb 8 ; 25/12/2016

```
; 15/04/2016
; TRDOS 386 (TRDOS v2.0)
; 80 interrupts
; 11/03/2015
; Interrupt Descriptor Table (20/08/2014)
```

```
idt:
;resb 64*8 ; INT 0 to INT 3Fh
; 15/04/2016
resb 80*8 ; INT 0 to INT 4Fh
```

idt_end:

;alignb 4

```
task_state_segment:
; 24/03/2015
tss.link: resw 1
        resw 1
; tss offset 4
tss.esp0: resd 1
tss.ss0:  resw 1
        resw 1
tss.esp1: resd 1
tss.ss1:  resw 1
        resw 1
tss.esp2: resd 1
tss.ss2:  resw 1
        resw 1
; tss offset 28
tss.CR3:  resd 1
tss.eip:  resd 1
tss.eflags: resd 1
; tss offset 40
tss.eax:  resd 1
tss.ecx:  resd 1
tss.edx:  resd 1
tss.ebx:  resd 1
tss.esp:  resd 1
tss.ebp:  resd 1
tss.esi:  resd 1
tss.edi:  resd 1
; tss offset 72
tss.ES:   resw 1
        resw 1
tss.CS:   resw 1
```

```

2726 00013806 <res 00000002>          resw 1
2727 00013808 <res 00000002>          resw 1
2728 0001380A <res 00000002>          resw 1
2729 0001380C <res 00000002>          resw 1
2730 0001380E <res 00000002>          resw 1
2731 00013810 <res 00000002>          resw 1
2732 00013812 <res 00000002>          resw 1
2733 00013814 <res 00000002>          resw 1
2734 00013816 <res 00000002>          resw 1
2735 00013818 <res 00000002>          resw 1
2736 0001381A <res 00000002>          resw 1
2737                                     ; tss offset 100
2738 0001381C <res 00000002>          resw 1
2739 0001381E <res 00000002>          resw 1
2740                                     ; tss offset 104
2741 tss_end:
2742
2743 00013820 <res 00000004>          k_page_dir: resd 1 ; Kernel's (System) Page Directory address
2744                                     ; (Physical address = Virtual address)
2745 00013824 <res 00000004>          memory_size: resd 1 ; memory size in pages
2746 00013828 <res 00000004>          free_pages: resd 1 ; number of free pages
2747 0001382C <res 00000004>          next_page:  resd 1 ; offset value in M.A.T. for
2748                                     ; first free page search
2749 00013830 <res 00000004>          last_page:  resd 1 ; offset value in M.A.T. which
2750                                     ; next free page search will be
2751                                     ; stopped after it. (end of M.A.T.)
2752 00013834 <res 00000004>          first_page: resd 1 ; offset value in M.A.T. which
2753                                     ; first free page search
2754                                     ; will be started on it. (for user)
2755 00013838 <res 00000004>          mat_size:  resd 1 ; Memory Allocation Table size in pages
2756
2757                                     ; 02/09/2014 (Retro UNIX 386 v1)
2758                                     ; 04/12/2013 (Retro UNIX 8086 v1)
2759 0001383C <res 00000002>          CRT_START:  resw 1          ; starting address in regen buffer
2760                                     ; NOTE: active page only
2761 0001383E <res 00000010>          CURSOR_POSN: resw 8 ; cursor positions for video pages
2762 ACTIVE_PAGE:
2763 0001384E <res 00000001>          ptty:         resb 1 ; current tty
2764                                     ; 01/07/2015 - 29/01/2016
2765 0001384F <res 00000001>          ccolor:       resb 1 ; current color attribute
2766                                     ; 26/10/2015
2767                                     ; 07/09/2014
2768 00013850 <res 00000014>          ttychr:       resw ntty+2 ; Character buffer (multiscreen)
2769
2770                                     ; 18/05/2015 (03/06/2013 - Retro UNIX 8086 v1 feature only!)
2771 00013864 <res 00000004>          p_time:        resd 1          ; present time (for systime & sysmdate)
2772
2773                                     ; 18/05/2015 (16/08/2013 - Retro UNIX 8086 v1 feature only !)
2774                                     ; (open mode locks for pseudo TTYs)
2775                                     ; [ major tty locks (return error in any conflicts) ]
2776 00013868 <res 00000014>          ttyl:         resw ntty+2 ; opening locks for TTYs.
2777
2778                                     ; 15/04/2015 (Retro UNIX 386 v1)
2779                                     ; 22/09/2013 (Retro UNIX 8086 v1)
2780 0001387C <res 0000000A>          wlist:        resb ntty+2 ; wait channel list (0 to 9 for TTYs)
2781                                     ; 15/04/2015 (Retro UNIX 386 v1)
2782                                     ; ; 12/07/2014 -> sp_init set comm. parameters as 0E3h
2783                                     ; ; 0 means serial port is not available
2784                                     ; ;comprm: ; 25/06/2014
2785 00013886 <res 00000001>          com1p:        resb 1          ; ;0E3h
2786 00013887 <res 00000001>          com2p:        resb 1          ; ;0E3h
2787
2788                                     ; 17/11/2015
2789                                     ; request for response (from the terminal)
2790 00013888 <res 00000002>          req_resp:      resw 1
2791                                     ; 07/11/2015
2792 0001388A <res 00000001>          ccomport:     resb 1 ; current COM (serial) port
2793                                     ; (0= COM1, 1= COM2)
2794
2795                                     ; 09/11/2015
2796 0001388B <res 00000001>          comqr:        resb 1 ; 'query or response' sign (u9.s, 'sndc')
2797                                     ; 07/11/2015
2798 0001388C <res 00000002>          rchar:        resw 1 ; last received char for COM 1 and COM 2
2799 0001388E <res 00000002>          schar:        resw 1 ; last sent char for COM 1 and COM 2
2800
2801                                     ; 22/08/2014 (RTC)
2802                                     ; (Packed BCD)
2802 00013890 <res 00000001>          time_seconds: resb 1
2803 00013891 <res 00000001>          time_minutes: resb 1
2804 00013892 <res 00000001>          time_hours:   resb 1
2805 00013893 <res 00000001>          date_wday:    resb 1
2806 00013894 <res 00000001>          date_day:     resb 1
2807 00013895 <res 00000001>          date_month:   resb 1
2808 00013896 <res 00000001>          date_year:    resb 1
2809 00013897 <res 00000001>          date_century: resb 1
2810
2811                                     ; 24/01/2016
2812 00013898 <res 00000004>          RTC_LH:        resd 1
2813 0001389C <res 00000001>          RTC_WAIT_FLAG: resb 1
2814 0001389D <res 00000001>          USER_FLAG:    resb 1
2815                                     ; 19/05/2016
2816                                     ; RTC_second:
2817 0001389E <res 00000001>          RTC_2Hz:      resb 1 ; from 2Hz interrupt to 1Hz timer event function
2818
2819 %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
21     <1> ; (Uninitialized Disk Parameters Data section for 'DISKIO.INC')
22     <1> ;
23 0001389F <res 00000001> <1> alignb 2
24     <1> ;
25     <1> ;-----
26     <1> ;     TIMER DATA AREA           :
27     <1> ;-----
28     <1> ;
29     <1> ; TIMER_LH:           ; 16/02/2015
30 000138A0 <res 00000002> <1> TIMER_LOW:      resw    1           ; LOW WORD OF TIMER COUNT
31 000138A2 <res 00000002> <1> TIMER_HIGH:     resw    1           ; HIGH WORD OF TIMER COUNT
32 000138A4 <res 00000001> <1> TIMER_OFI:      resb    1           ; TIMER HAS ROLLED OVER SINCE LAST READ
33     <1> ;
34     <1> ;-----
35     <1> ;     DISKETTE DATA AREAS       :
36     <1> ;-----
37     <1> ;
38 000138A5 <res 00000001> <1> SEEK_STATUS: resb    1
39 000138A6 <res 00000001> <1> MOTOR_STATUS: resb    1
40 000138A7 <res 00000001> <1> MOTOR_COUNT: resb    1
41 000138A8 <res 00000001> <1> DSKETTE_STATUS: resb    1
42 000138A9 <res 00000007> <1> NEC_STATUS:  resb    7
43     <1> ;
44     <1> ;-----
45     <1> ;     ADDITIONAL MEDIA DATA     :
46     <1> ;-----
47     <1> ;
48 000138B0 <res 00000001> <1> LATESTATE:   resb    1
49 000138B1 <res 00000001> <1> HF_STATUS:   resb    1
50 000138B2 <res 00000001> <1> HF_ERROR:    resb    1
51 000138B3 <res 00000001> <1> HF_INT_FLAG: resb    1
52 000138B4 <res 00000001> <1> HF_CNTRL:    resb    1
53 000138B5 <res 00000004> <1> DSK_STATE:   resb    4
54 000138B9 <res 00000002> <1> DSK_TRK:     resb    2
55     <1> ;
56     <1> ;-----
57     <1> ;     FIXED DISK DATA AREAS     :
58     <1> ;-----
59     <1> ;
60 000138BB <res 00000001> <1> DISK_STATUS1: resb    1           ; FIXED DISK STATUS
61 000138BC <res 00000001> <1> HF_NUM:       resb    1           ; COUNT OF FIXED DISK DRIVES
62 000138BD <res 00000001> <1> CONTROL_BYTE: resb    1           ; HEAD CONTROL BYTE
63     <1> ;@PORT_OFF  resb    1           ; RESERVED (PORT OFFSET)
64     <1> ;port1_off  resb    1           ; Hard disk controller 1 - port offset
65     <1> ;port2_off  resb    1           ; Hard idsk controller 2 - port offset
66     <1> ;
67 000138BE <res 00000002> <1> alignb 4
68     <1> ;
69     <1> ;HF_TBL_VEC: resd    1           ; Primary master disk param. tbl. pointer
70     <1> ;HF1_TBL_VEC: resd    1           ; Primary slave disk param. tbl. pointer
71     <1> ; HF_TBL_VEC: ; 22/12/2014
72 000138C0 <res 00000004> <1> HDPM_TBL_VEC: resd    1           ; Primary master disk param. tbl. pointer
73 000138C4 <res 00000004> <1> HDPS_TBL_VEC: resd    1           ; Primary slave disk param. tbl. pointer
74 000138C8 <res 00000004> <1> HDMS_TBL_VEC: resd    1           ; Secondary master disk param. tbl. pointer
75 000138CC <res 00000004> <1> HDSS_TBL_VEC: resd    1           ; Secondary slave disk param. tbl. pointer
76     <1> ;
77     <1> ; 03/01/2015
78 000138D0 <res 00000001> <1> LBAMode:      resb    1
79     <1> ;
80     <1> ; *****
2820
2821     ;;; Real Mode Data (10/07/2015 - BSS)
2822
2823     ;alignb 2
2824
2825     ; 10/01/2016
2826     %include 'trdoskx.s' ; UNINITIALIZED KERNEL (Logical Drive & FS) DATA
1      <1> ; *****
2      <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - UNINITIALIZED DATA : trdoskx.s
3      <1> ; -----
4      <1> ; Last Update: 28/02/2017
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> ; DRV_INIT.ASM [26/09/2009] Last Update: 07/08/2011
14     <1> ; MAINPROG.ASM [17/01/2004] Last Update: 09/11/2011
15     <1> ; DIR.ASM [17/01/2004] Last Update: 09/10/2011
16     <1> ; CMD_INTR.ASM [29/01/2005] Last update: 09/11/2011
17     <1> ; DRV_FAT.ASM [07/07/2009] Last update: 21/08/2011
18     <1> ;
19 000138D1 <res 00000003> <1> alignb 4
20     <1> ;
21     <1> ; MAINPROG.ASM
22 000138D4 <res 00000004> <1> MainProgCfg_FileSize: resd 1 ; 14/04/2016
23 000138D8 <res 00000004> <1> MainProgCfg_LineOffset: resd 1 ; 14/04/2016

```

```

24                                     <1>
25 000138DC <res 00000004>           <1> Current_VolSerial: resd 1
26                                     <1>
27 000138E0 <res 00000004>           <1> Current_Dir_FCluster: resd 1
28                                     <1>
29 000138E4 <res 00000001>           <1> Current_Dir_Level: resb 1
30 000138E5 <res 00000001>           <1> Current_FATType: resb 1
31 000138E6 <res 00000001>           <1> Current_Drv: resb 1
32 000138E7 <res 00000001>           <1> Current_Dir_Drv:   resb 1 ; '?'
33 000138E8 <res 00000001>           <1>                               resb 1 ; ':'
34 000138E9 <res 00000001>           <1> Current_Dir_Root:   resb 1 ; '/'
35 000138EA <res 0000005A>           <1> Current_Directory: resb 90
36 00013944 <res 00000001>           <1> End_Of_Current_Dir_Str: resb 1
37 00013945 <res 00000001>           <1> Current_Dir_StrLen: resb 1
38                                     <1>
39 00013946 <res 00000001>           <1> CursorColumn:      resb 1
40 00013947 <res 00000001>           <1> CmdArgStart:       resb 1
41                                     <1>
42                                     <1> ; 03/02/2016
43 00013948 <res 0000004E>           <1> Remark:              resb 78
44                                     <1>
45 00013996 <res 00000050>           <1> CommandBuffer:      resb 80
46                                     <1>
47 000139E6 <res 00000100>           <1> TextBuffer:   resb 256
48                                     <1>
49                                     <1> MasterBootBuff:
50 00013AE6 <res 000001BE>           <1> MasterBootCode: resb 1BEh
51 00013CA4 <res 00000040>           <1> PartitionTable: resb 64
52 00013CE4 <res 00000002>           <1> MBIDCode: resw 1
53                                     <1>
54                                     <1> PTable_Buffer:
55 00013CE6 <res 00000040>           <1> PTable_hd0: resb 64
56 00013D26 <res 00000040>           <1> PTable_hd1: resb 64
57 00013D66 <res 00000040>           <1> PTable_hd2: resb 64
58 00013DA6 <res 00000040>           <1> PTable_hd3: resb 64
59 00013DE6 <res 00000040>           <1> PTable_ep0: resb 64
60 00013E26 <res 00000040>           <1> PTable_ep1: resb 64
61 00013E66 <res 00000040>           <1> PTable_ep2: resb 64
62 00013EA6 <res 00000040>           <1> PTable_ep3: resb 64
63                                     <1>
64 00013EE6 <res 00000001>           <1> scount:             resb 1 ; 16/05/2016 (diskio.s,'int33h:')
65 00013EE7 <res 00000001>           <1> HD_LBA_yes: resb 1
66 00013EE8 <res 00000001>           <1> PP_Counter: resb 1
67 00013EE9 <res 00000001>           <1> EP_Counter: resb 1
68                                     <1>
69 00013EEA <res 00000004>           <1> EP_StartSector: resd 1
70 00013EEE <res 00000004>           <1>                               resd 1
71 00013EF2 <res 00000004>           <1>                               resd 1
72 00013EF6 <res 00000004>           <1>                               resd 1
73                                     <1>
74 00013EFA <res 00000200>           <1> DOSBootSectorBuff: resb 512
75                                     <1>
76                                     <1> FAT_BuffDescriptor:
77 000140FA <res 00000004>           <1> FAT_CurrentCluster: resd 1
78 000140FE <res 00000001>           <1> FAT_BuffValidData: resb 1
79 000140FF <res 00000001>           <1> FAT_BuffDrvName: resb 1
80 00014100 <res 00000002>           <1> FAT_BuffOffset: resw 1
81 00014102 <res 00000004>           <1> FAT_BuffSector: resd 1
82                                     <1>
83 00014106 <res 00000004>           <1> FAT_ClusterCounter: resd 1
84 0001410A <res 00000004>           <1> LastCluster: resd 1
85                                     <1>
86                                     <1> ; 16/05/2016
87                                     <1> ;; 18/03/2016 (TRDOS v2.0)
88                                     <1> ;ClusterBuffer_Valid: resb 1
89                                     <1>
90                                     <1> Dir_BuffDescriptor:
91 0001410E <res 00000001>           <1> DirBuff_DRV: resb 1
92 0001410F <res 00000001>           <1> DirBuff_FATType: resb 1
93 00014110 <res 00000001>           <1> DirBuff_ValidData: resb 1
94 00014111 <res 00000002>           <1> DirBuff_CurrentEntry: resw 1
95 00014113 <res 00000002>           <1> DirBuff_LastEntry: resw 1
96 00014115 <res 00000004>           <1> DirBuff_Cluster: resd 1
97 00014119 <res 00000002>           <1> DirBuffer_Size: resw 1
98                                     <1> ;DirBuff_EntryCounter: resw 1
99                                     <1>
100                                    <1> ; 01/02/2016
101                                    <1> ; these are on (real mode) segment 8000h and later
102                                    <1> ; FAT_Buffer:      resb 1536 ; 3 sectors
103                                    <1> ; Dir_Buffer:      resb 512*32
104                                    <1> ; Logical_DOSDisks: resb 6656 ; 26 * 256 bytes
105                                    <1>
106                                    <1> ; 18/01/2016
107                                    <1>
108 0001411B <res 00000004>           <1> FreeClusterCount: resd 1
109                                    <1>
110 0001411F <res 00000004>           <1> VolSize_Unit1:   resd 1
111 00014123 <res 00000004>           <1> VolSize_Unit2:   resd 1
112                                    <1>
113 00014127 <res 00000004>           <1> Vol_Tot_Sec_Str_Start: resd 1
114 0001412B <res 0000000A>           <1> Vol_Tot_Sec_Str:   resb 10
115 00014135 <res 00000001>           <1> Vol_Tot_Sec_Str_End: resb 1
116 00014136 <res 00000001>           <1> resb 1
117 00014137 <res 00000004>           <1> Vol_Free_Sectors_Str_Start: resd 1
118 0001413B <res 0000000A>           <1> Vol_Free_Sectors_Str: resb 10
119 00014145 <res 00000001>           <1> Vol_Free_Sectors_Str_End: resb 1
120                                    <1>
121                                    <1> ; 10/02/2016
122 00014146 <res 00000001>           <1> RUN_CDRV: resb 1 ; CMD_INTR.ASM ; 09/11/2011
123                                    <1>
124                                    <1> ; 24/01/2016
125 00014147 <res 00000080>           <1> PATH_Array:      resb 128 ; DIR.ASM ; 09/10/2011

```

```

126
127 000141C7 <res 00000004>
128 000141CB <res 00000001>
129 000141CC <res 00000001>
130
131 000141CD <res 00000002>
132 000141CF <res 00000004>
133 000141D3 <res 00000002>
134
135 000141D5 <res 00000001>
136
137 000141D6 <res 00000002>
138
139
140 000141D8 <res 00000001>
141
142
143
144 000141D9 <res 00000001>
145
146
147 000141DA <res 00000001>
148 000141DB <res 00000001>
149 000141DC <res 00000004>
150
151
152 000141E0 <res 00000002>
153 000141E2 <res 00000002>
154 000141E4 <res 00000001>
155
156 000141E5 <res 00000001>
157 000141E6 <res 00000001>
158 000141E7 <res 00000001>
159 000141E8 <res 00000084>
160
161
162 0001426C <res 00000001>
163 0001426D <res 00000001>
164
165
166 0001426E <res 0000000D>
167
168
169 0001427B <res 0000000D>
170
171
172
173 00014288 <res 00000002>
174
175
176
177
178 0001428A <res 00000001>
179 0001428B <res 00000041>
180 000142CC <res 0000000D>
181
182 000142D9 <res 00000001>
183
184
185 000142DA <res 00000002>
186 000142DC <res 00000020>
187 000142FC <res 00000004>
188 00014300 <res 00000004>
189 00014304 <res 00000002>
190 00014306 <res 00000002>
191 00014308 <res 00000002>
192
193 0001430A <res 00000004>
194 0001430E <res 00000004>
195
196
197 00014312 <res 00000002>
198 00014314 <res 00000002>
199 00014316 <res 00000004>
200 0001431A <res 00000004>
201 0001431E <res 0000000A>
202 00014328 <res 00000001>
203
204 00014329 <res 00000001>
205
206 0001432A <res 00000002>
207
208 0001432C <res 00000004>
209 00014330 <res 00000004>
210 00014334 <res 00000004>
211 00014338 <res 00000004>
212 0001433C <res 00000004>
213 00014340 <res 00000002>
214 00014342 <res 00000002>
215 00014344 <res 00000001>
216
217 00014345 <res 00000003>
218
219 00014348 <res 00000004>
220
221
222 0001434C <res 00000004>
223 00014350 <res 00000002>
224 00014352 <res 00000001>
225 00014353 <res 00000001>
226
227

<1> ; 06/02/2016
<1> CCD_DriveDT: resd 1 ; DIR.ASM ; (word)
<1> CCD_Level:   resb 1 ; DIR.ASM
<1> Last_Dir_Level:   resb 1 ; DIR.ASM
<1> ;
<1> CDLF_AttributesMask: resw 1 ; DIR.ASM
<1> CDLF_FNAddress:   resd 1 ; DIR.ASM (word)
<1> CDLF_DEType: resw 1 ; DIR.ASM
<1> ;
<1> CD_COMMAND:   resb 1 ; DIR.ASM
<1>
<1> alignb 4
<1>
<1> ; 29/01/2016
<1> Program_Exit:   resb 1 ; CMD_INTR.ASM ; 09/11/2011
<1>
<1> ;alignb 4
<1> ; 23/02/2016
<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
<1> retry_count:   resb 1 ; DISK_IO.ASM ; 20/07/2011 (CHS_RetryCount)
<1> disk_rw_err:   resb 1 ; DISK_IO.ASM ; (Disk_IO_err_code)
<1> sector_count: resd 1 ; DISK_IO.ASM ; (Disk_RW_SectorCount)
<1>
<1> ; 06/02/2016 (long name)
<1> FDE_AttrMask:   resw 1 ; DIR.ASM
<1> AmbiguousFileName: resw 1 ; DIR.ASM
<1> PreviousAttr:   resb 1 ; DIR.ASM
<1> ;
<1> LongNameFound:   resb 1 ; DIR.ASM
<1> LFN_EntryLength: resb 1 ; DIR.ASM
<1> LFN_CheckSum:   resb 1 ; DIR.ASM
<1> LongFileName:   resb 132 ; DIR.ASM
<1>
<1> ;PATH_Array_Ptr: resw 1 ; DIR.ASM
<1> PATH_CDLevel:   resb 1 ; DIR.ASM
<1> PATH_Level:   resb 1 ; DIR.ASM
<1>
<1> ; 07/02/2016
<1> Dir_File_Name:   resb 13 ; DIR.ASM ; 09/10/2011
<1>
<1> ; 10/02/2016
<1> Dir_Entry_Name:   resb 13 ; DIR.ASM
<1>
<1> alignb 2
<1>
<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
<1> FindFile_Drv:   resb 1
<1> FindFile_Directory: resb 65
<1> FindFile_Name:   resb 13
<1> FindFile_LongNameEntryLength:
<1> FindFile_LongNameYes: resb 1 ; Sign for longname procedures
<1> ;Above 80 bytes form
<1> ;TR-DOS Source/Destination File FullName Format/Structure
<1> FindFile_AttributesMask: resw 1
<1> FindFile_DirEntry:   resb 32
<1> FindFile_DirFirstCluster: resd 1
<1> FindFile_DirCluster:   resd 1
<1> FindFile_DirEntryNumber: resw 1
<1> FindFile_MatchCounter: resw 1
<1> FindFile_Reserved:   resw 1 ; 06/03/2016
<1>
<1> First_Path_Pos: resd 1 ; DIR.ASM ; 09/10/2011
<1> Last_Slash_Pos: resd 1 ; DIR.ASM
<1>
<1> ; 10/02/2016
<1> File_Count:   resw 1 ; DIR.ASM ; 09/10/2011
<1> Dir_Count:   resw 1
<1> Total_FSize: resd 1
<1> TFS_Dec_Begin: resd 1
<1>   resb 10
<1> TFS_Dec_End:   resb 1
<1>
<1> PrintDir_RowCounter: resb 1
<1>
<1> alignb 4
<1> ; 15/02/2015 ('show' command variables)
<1> Show_FDT:   resd 1
<1> Show_LDDDT: resd 1
<1> Show_Cluster:   resd 1
<1> Show_FileSize:   resd 1
<1> Show_FilePointer: resd 1
<1> Show_ClusterPointer: resw 1
<1> Show_ClusterSize: resw 1
<1> Show_RowCount:   resb 1
<1>
<1> alignb 4
<1> ; 21/02/2016
<1> DelFile_FNPointer: resd 1 ; ; CMD_INTR.ASM (word) ; 09/11/2011
<1> ; 27/02/2016
<1> ; DIR.ASM (09/10/2011)
<1> DelFile_FCluster: resd 1
<1> DelFile_EntryCounter: resw 1
<1> DelFile_LNEL:   resb 1
<1> resb 1
<1>
<1> ; DIR.ASM

```

```

228 00014354 <res 00000004> <1> mkdir_DirName_Offset: resd 1
229 00014358 <res 00000004> <1> mkdir_FFCluster: resd 1
230 0001435C <res 00000004> <1> mkdir_LastDirCluster: resd 1
231 00014360 <res 00000004> <1> mkdir_FreeSectors: resd 1
232 00014364 <res 00000002> <1> mkdir_attrb: resw 1
233 00014366 <res 00000001> <1> mkdir_SecPerClust: resb 1
234 00014367 <res 00000001> <1> mkdir_add_new_cluster: resb 1
235 00014368 <res 0000000D> <1> mkdir_Name: resb 13
236 00014375 <res 00000002> <1> resw 1 ; 01/03/2016
237 <1> ; 27/02/2016
238 00014377 <res 00000001> <1> Rmdir_MultiClusters: resb 1
239 00014378 <res 00000004> <1> Rmdir_DirEntryOffset: resd 1 ; 01/03/2016 (word -> dword)
240 0001437C <res 00000004> <1> Rmdir_ParentDirCluster: resd 1
241 00014380 <res 00000004> <1> Rmdir_DirLastCluster: resd 1
242 00014384 <res 00000004> <1> Rmdir_PreviousCluster: resd 1
243 <1> ; 22/02/2016
244 00014388 <res 00000001> <1> UPDLMDT_CDirLevel: resb 1
245 00014389 <res 00000004> <1> UPDLMDT_CDirFCluster: resd 1
246 <1>
247 0001438D <res 00000003> <1> alignb 4
248 <1> ; DRV_FAT.ASM ; 21/08/2011
249 00014390 <res 00000004> <1> gffc_next_free_cluster: resd 1
250 00014394 <res 00000004> <1> gffc_first_free_cluster: resd 1
251 00014398 <res 00000004> <1> gffc_last_free_cluster: resd 1
252 <1>
253 <1> ;29/04/2016
254 <1> Cluster_Index: ; resd 1
255 <1> ; 22/02/2016
256 0001439C <res 00000004> <1> ClusterValue: resd 1
257 <1> ; 04/03/2016
258 000143A0 <res 00000001> <1> Attributes: resb 1
259 <1> ;;CFS_error: resb 1 ;; 01/03/2016
260 000143A1 <res 00000001> <1> resb 1
261 000143A2 <res 00000001> <1> CFS_OPType: resb 1
262 000143A3 <res 00000001> <1> CFS_Drv: resb 1
263 000143A4 <res 00000004> <1> CFS_CC: resd 1
264 000143A8 <res 00000004> <1> CFS_FAT32FSINFOSEC: resd 1
265 000143AC <res 00000004> <1> CFS_FAT32FC: resd 1
266 <1>
267 <1> ; 27/02/2016
268 <1> ;alignb 4
269 000143B0 <res 00000004> <1> glc_prevcluster: resd 1 ; DRV_FAT.ASM (21/08/2011)
270 <1> ; 22/10/2016
271 000143B4 <res 00000004> <1> glc_index: resd 1 ; Last Cluster Index (22/10/2016)
272 <1>
273 <1> ; DIR.ASM
274 000143B8 <res 00000002> <1> DLN_EntryNumber: resw 1
275 000143BA <res 00000001> <1> DLN_40h: resb 1
276 <1> ; 28/02/2016
277 000143BB <res 00000001> <1> TCC_FATErr: resb 1 ; DRV_FAT.ASM
278 <1>
279 <1> alignb 4
280 <1> ; DIR.ASM (09/10/2011)
281 000143BC <res 00000002> <1> LCDE_EntryIndex: resw 1 ; LCDE_EntryOffset
282 000143BE <res 00000002> <1> LCDE_ClusterSN: resw 1
283 000143C0 <res 00000004> <1> LCDE_Cluster: resd 1
284 000143C4 <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 000143C8 <res 00000004> <1> SourceFilePath: resd 1
290 000143CC <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 000143D0 <res 00000001> <1> SourceFile_Drv: resb 1
297 000143D1 <res 00000041> <1> SourceFile_Directory: resb 65
298 00014412 <res 0000000D> <1> SourceFile_Name: resb 13
299 <1> SourceFile_LongNameEntryLength:
300 0001441F <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 00014420 <res 00000002> <1> SourceFile_AttributesMask: resw 1
304 00014422 <res 00000020> <1> SourceFile_DirEntry: resb 32
305 00014442 <res 00000004> <1> SourceFile_DirFirstCluster: resd 1
306 00014446 <res 00000004> <1> SourceFile_DirCluster: resd 1
307 0001444A <res 00000002> <1> SourceFile_DirEntryNumber: resw 1
308 0001444C <res 00000002> <1> SourceFile_MatchCounter: resw 1
309 <1> ; 16/03/2016
310 0001444E <res 00000001> <1> SourceFile_SecPerClust: resb 1
311 0001444F <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 00014450 <res 00000001> <1> DestinationFile_Drv: resb 1
316 00014451 <res 00000041> <1> DestinationFile_Directory: resb 65
317 00014492 <res 0000000D> <1> DestinationFile_Name: resb 13
318 <1> DestinationFile_LongNameEntryLength:
319 0001449F <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 000144A0 <res 00000002> <1> DestinationFile_AttributesMask: resw 1
323 000144A2 <res 00000020> <1> DestinationFile_DirEntry: resb 32
324 000144C2 <res 00000004> <1> DestinationFile_DirFirstCluster: resd 1
325 000144C6 <res 00000004> <1> DestinationFile_DirCluster: resd 1
326 000144CA <res 00000002> <1> DestinationFile_DirEntryNumber: resw 1
327 000144CC <res 00000002> <1> DestinationFile_MatchCounter: resw 1
328 <1> ; 16/03/2016
329 000144CE <res 00000001> <1> DestinationFile_SecPerClust: resb 1

```



```

330 000144CF <res 00000001> <1> DestinationFile_Reserved: resb 1
331 <1> ; Above is 128 bytes
332 <1>
333 <1> ; 24/04/2016
334 000144D0 <res 00000002> <1> resw 1
335 <1>
336 <1> ; 10/03/2016
337 <1> ; FILE.ASM
338 000144D2 <res 00000001> <1> move_cmd_phase: resb 1
339 000144D3 <res 00000001> <1> msftdf_sf_df_drv: resb 1
340 000144D4 <res 00000004> <1> msftdf_drv_offset: resd 1
341 <1>
342 <1> ; 11/03/2016
343 <1> ; DRV_FAT.ASM (21/08/2011)
344 000144D8 <res 00000004> <1> FAT_anc_LCluster: resd 1
345 000144DC <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 000144E0 <res 00000004> <1> mem_ipg_count: resd 1 ; page count (for contiguous allocation)
353 000144E4 <res 00000004> <1> mem_pg_count: resd 1 ; page count (for count down)
354 000144E8 <res 00000004> <1> mem_aperture: resd 1 ; contiguous free pages (current)
355 000144EC <res 00000004> <1> mem_max_aperture: resd 1 ; maximum value of contiguous free pages
356 000144F0 <res 00000004> <1> mem_pg_pos: resd 1 ; mem. position (page #) of current aperture
357 000144F4 <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 000144F8 <res 00000001> <1> copy_cmd_phase: resb 1
362 000144F9 <res 00000001> <1> csftdf_rw_err: resb 1
363 000144FA <res 00000001> <1> DestinationFileFound: resb 1
364 000144FB <res 00000001> <1> csftdf_cdrv: resb 1
365 000144FC <res 00000004> <1> csftdf_filesize: resd 1
366 <1> ; TRDOS386 (TRDOS v2.0)
367 00014500 <res 00000004> <1> csftdf_sf_mem_addr: resd 1
368 00014504 <res 00000004> <1> csftdf_sf_mem_bsize: resd 1
369 <1> ;
370 <1>
371 00014508 <res 00000004> <1> csftdf_sf_cluster: resd 1 ; 16/03/2016
372 0001450C <res 00000004> <1> csftdf_df_cluster: resd 1
373 <1> ; 16/03/2016
374 00014510 <res 00000004> <1> csftdf_r_size: resd 1
375 00014514 <res 00000004> <1> csftdf_w_size: resd 1
376 00014518 <res 00000004> <1> csftdf_sf_rbytes: resd 1
377 0001451C <res 00000004> <1> csftdf_df_wbytes: resd 1
378 00014520 <res 00000001> <1> csftdf_percentage: resb 1
379 <1> ; 17/03/2016
380 00014521 <res 00000001> <1> csftdf_videopage: resb 1
381 00014522 <res 00000002> <1> csftdf_cursorpos: resw 1
382 00014524 <res 00000004> <1> csftdf_sf_drv_dt: resd 1
383 00014528 <res 00000004> <1> csftdf_df_drv_dt: resd 1
384 <1>
385 <1> ; 21/03/2016
386 <1> ; 20/03/2016
387 <1> ; FILE.ASM
388 0001452C <res 00000004> <1> createfile_Name_Offset: resd 1
389 00014530 <res 00000004> <1> createfile_FreeSectors: resd 1
390 00014534 <res 00000004> <1> createfile_size: resd 1
391 00014538 <res 00000004> <1> createfile_FFCluster: resd 1 ; 11/03/2016
392 0001453C <res 00000004> <1> createfile_LastDirCluster: resd 1
393 00014540 <res 00000004> <1> createfile_Cluster: resd 1
394 00014544 <res 00000004> <1> createfile_PCluster: resd 1
395 00014548 <res 00000001> <1> createfile_attrib: resb 1
396 00014549 <res 00000001> <1> createfile_SecPerClust: resb 1
397 0001454A <res 00000002> <1> createfile_DirIndex: resw 1
398 0001454C <res 00000004> <1> createfile_CCount: resd 1
399 00014550 <res 00000002> <1> createfile_BytesPerSec: resw 1 ; 23/03/2016
400 00014552 <res 00000001> <1> createfile_wfc: resb 1
401 00014553 <res 00000001> <1> createfile_UpdatePDir: resb 1 ; 31/03/2016
402 <1>
403 <1> ;alignb 4
404 <1>
405 <1> ; 11/04/2016
406 00014554 <res 00000002> <1> env_var_length: resw 1
407 <1>
408 00014556 <res 00000002> <1> alignb 4
409 <1>
410 <1> ; 25/04/2016
411 00014558 <res 00000001> <1> readi.valid: resb 1 ; valid data (>0 = valid for readi)
412 00014559 <res 00000001> <1> readi.drv: resb 1 ; drive number (0, 1,2,3,4..)
413 0001455A <res 00000001> <1> readi.spc: resb 1 ; sectors per cluster for 'readi' drive
414 0001455B <res 00000001> <1> readi.s_index: resb 1 ; sector index in current cluster (buffer)
415 0001455C <res 00000004> <1> readi.sector: resd 1 ; current disk sector
416 00014560 <res 00000002> <1> readi.bpc: resw 1 ; bytes per cluster - 1
417 00014562 <res 00000002> <1> readi.offset: resw 1 ; byte offset in cluster buffer
418 00014564 <res 00000004> <1> readi.cluster: resd 1 ; current cluster number
419 00014568 <res 00000004> <1> readi.c_index: resd 1 ; cluster index of the current cluster (0,1,2,3..)
420 0001456C <res 00000004> <1> readi.fc_lust: resd 1 ; first cluster of the current cluster
421 00014570 <res 00000004> <1> readi.fs_index: resd 1 ; sector index in disk/file section (for Singlix FS)
422 <1> ;readi.buffer: resd 1 ; readi sector buffer address
423 <1>
424 <1> ;alignb 4
425 <1>
426 00014574 <res 00000001> <1> writei.valid: resb 1 ; valid data (>0 = valid for writei)
427 00014575 <res 00000001> <1> writei.drv: resb 1 ; drive number (0, 1,2,3,4..)
428 00014576 <res 00000001> <1> writei.spc: resb 1 ; sectors per cluster for 'writei' drive
429 00014577 <res 00000001> <1> writei.s_index: resb 1 ; sector index in current cluster (buffer)
430 00014578 <res 00000004> <1> writei.sector: resd 1 ; current disk sector
431 0001457C <res 00000002> <1> writei.bpc: resw 1 ; bytes per cluster - 1

```

```

432 0001457E <res 00000002> <1> writei.offset: resw 1 ; byte offset in cluster buffer
433 00014580 <res 00000004> <1> writei.cluster: resd 1 ; current cluster number
434 00014584 <res 00000004> <1> writei.c_index: resd 1 ; cluster index of the current cluster (0,1,2,3..)
435 00014588 <res 00000004> <1> writei.fclust: resd 1 ; first cluster of the current cluster
436 0001458C <res 00000004> <1> writei.fs_index: resd 1 ; sector index in disk/file section (for Singlix FS)
437 <1> ;writei.buffer: resd 1 ; writei sector buffer address
438 00014590 <res 00000004> <1> writei.lclust: resd 1 ; writei last cluster (mget_w) ; 23/10/2016
439 00014594 <res 00000004> <1> writei.l_index: resd 1 ; writei last cluster index (mget_w) ; 23/10/2016
440 00014598 <res 00000001> <1> writei.ofn: resb 1 ; open file number (to be written) ; 23/10/2016
441 <1>
442 00014599 <res 00000003> <1> alignb 4
443 <1>
444 <1> ; 29/04/2016
445 0001459C <res 00000004> <1> Run_CDirFC: resd 1
446 000145A0 <res 00000001> <1> Run_Auto_Path: resb 1
447 000145A1 <res 00000001> <1> Run_Manual_Path: resb 1 ; 0 -> auto path sequence needed
448 000145A2 <res 00000001> <1> EXE_ID: resb 1
449 000145A3 <res 00000001> <1> EXE_dot: resb 1
450 <1>
451 <1> ; 06/05/2016
452 000145A4 <res 00000004> <1> mainprog_return_addr: resd 1
453 000145A8 <res 00000004> <1> last_error: resd 1 ; this will be used to return error code to MainProg
454 <1> ; 'lasterror' keyword will be used later to get the
455 <1> ; last error code/number/status.
456 <1> ; 12/05/2016
457 000145AC <res 00000004> <1> video_eax: resd 1 ; eax return value of video function
458 <1>
459 <1> ; 01/06/2016
460 000145B0 <res 00000004> <1> user_buffer: resd 1 ; 'diskio.s' (INT 33h, Function 08h, floppy disk type)
461 <1>
462 <1> ; 21/05/2016 - TRDOS 386 ('swap/switch', 'rswap', [u.pri])
463 000145B4 <res 00000001> <1> priority: resb 1 ; running priority level of process (0,1,2)
464 <1> ; (run queue which is process comes from)
465 <1> ; 22/05/2016 - TRDOS 386 ('set_run_sequence', 'rtc_int', 'u_timer')
466 000145B5 <res 00000001> <1> p_change: resb 1 ; process change status (for timer events)
467 <1> ; 23/05/2016 - TRDOS 386 ('clock')
468 000145B6 <res 00000001> <1> multi_tasking: resb 1 ; Multi Tasking status (0 = disabled, >0 = enabled)
469 <1> ; (EBX will return with user buffer addr or disk type)
470 <1> ; 07/06/2016
471 000145B7 <res 00000001> <1> timer_events: resb 1 ; number of (active) timer events, <= 16
472 <1>
473 <1> ; 24/06/2016
474 000145B8 <res 00000001> <1> w_str_cmd: resb 1 ; WRITE_STRING command (0,1,2,3) ; video.s
475 000145B9 <res 00000001> <1> p_crt_mode: resb 1 ; previous video mode (=3 or 0), backup mark/sign
476 <1> ; 26/06/2016
477 000145BA <res 00000001> <1> p_crt_page: resb 1 ; previous active page (for 'set_mode')
478 <1> ; 04/07/2016
479 000145BB <res 00000001> <1> noclearmem: resb 1 ; if set, 'SET MODE' (INT 31h) function (AH = 4)
480 <1> ; will not clear the video memory
481 <1> ; (usable for graphics modes only)
482 <1> alignb 2
483 000145BC <res 00000002> <1> CRT_LEN: resw 1 ; length of regen buffer in bytes
484 000145BE <res 00000010> <1> cursor_pposn: resw 8 ; cursor positions backup
485 <1>
486 <1> ; 10/07/2016 ('VGA_FONT_SETUP', INT 43H address for x86 real mode bios)
487 000145CE <res 00000004> <1> VGA_INT43H: resd 1 ; 0 = default (not configured by user)
488 <1> ; 0FFFFFFFh = user defined fonts
489 <1> ; address:
490 <1> ; vgafont8
491 <1> ; vgafont16
492 <1> ; vgafont14
493 <1>
494 <1> ; 25/07/2016
495 000145D2 <res 00000001> <1> VGA_MTYPE: resb 1 ; 0=CTEXT,1=MTEXT,2=CGA,3=PLANAR1,4=PLANAR4,5=LINEAR
496 <1>
497 <1> ; 23/10/2016
498 000145D3 <res 00000001> <1> setfmod resb 1 ; update last modification date&time sign (if >0)
499 <1> ; (it is Open File Number + 1, if > 0)
500 <1> alignb 4
501 <1>
502 <1> ; 16/10/2016
503 000145D4 <res 00000004> <1> FFF_UBuffer: resd 1 ; User's buffer address for FFF & FNF system calls
504 <1> ; 15/10/2016
505 000145D8 <res 00000001> <1> FFF_Valid: resb 1 ; Find First File Structure validation byte
506 <1> ; 0 = invalid (Find Next File can't use FFF struct)
507 <1> ; >0 = valid, return type for FFF and Find Next File
508 <1> ; 24 = basic parameters, 24 bytes
509 <1> ; 128 = entire FFF structure/table, 128 bytes
510 <1> ; 16/10/2016 (FFF_Attrib: resw 1)
511 000145D9 <res 00000001> <1> FFF_Attrib: resb 1 ; Find First File attributes for Find Next File (LB)
512 000145DA <res 00000001> <1> FFF_RType: resb 1 ; FFF return type (0 = Basic, >0 = complete) (HB)
513 <1> ; 16/10/2016 - 05/10/2016 (Set Working Path)
514 000145DB <res 00000001> <1> SWP_inv_fname: resb 1 ; Set Working Path - Invalid File Name
515 000145DC <res 00000002> <1> SWP_Mode: resw 1 ; Set Working Path - Mode
516 000145DE <res 00000001> <1> SWP_DRV: resb 1 ; Set Working Path - Drive
517 000145DF <res 00000001> <1> SWP_DRV_chg: resb 1 ; Set Working Path - Drive Change
518 <1>
519 <1> ; 27/02/2017
520 000145E0 <res 00000001> <1> fpready: resb 1 ; '80387 fpu is ready' flag
521 <1>
522 <1> ; 08/10/2016
523 000145E1 <res 00000009> <1> device_name: resb 9 ; capitalized (and zero padded) device canem
524 <1> ; (example: "TTY0",0,0,0,0,0)
525 <1>
526 000145EA <res 00000002> <1> alignb 4
527 <1>
528 <1> ; 08/10/2016
529 <1> ; 07/10/2016
530 <1> ; Table of kernel devices (which do not use installable device drivers)
531 <1> ; has been coded into KERNEL (trdosk9.s)
532 <1> ; 07/10/2016
533 <1> ; 8 installable device drivers available to install (NUMIDEV)

```

```

534 000145EC <res 00000020> <1> IDEV_PGDIR: resd NUMIDEV
535 <1> ; Page directories of installable device drivers
536 <1> ;
537 <1> ; Note: Virtual start address is always 400000h
538 <1> ; (end of the 1st 4MB). [org 400000h]
539 <1> ; Segments: KCODE, KDATA
540 <1> ; Method: call 400000h (after changing page dir)
541 <1> ; Query code located at the start (400000h).
542 <1> ; Query code returns with
543 <1> ;   eax = device type and driver version
544 <1> ;       AL = Device Type minor
545 <1> ;       AH = Device Type major
546 <1> ;       Byte 16-23 : Version minor
547 <1> ;       Byte 24-31 : Version major - 1
548 <1> ;       (0:0 -> 1.0)
549 <1> ;   ebx = initialization code address
550 <1> ;   ecx = configuration table address
551 <1> ;   edx = description table address
552 <1> ;   esi = device (default) name address (ASCIIIZ)
553 <1> ;       (name has "/DEV/" prefix)
554 <1> ;   edi = dispatch table address
555 <1> ;       (for calling kernel-device functions)
556 <1> ;   ebp = address table address
557 <1> ; Initialization code returns with
558 <1> ;   eax = open code address
559 <1> ;   ecx = close code address
560 <1> ;   ebx = read code address
561 <1> ;   edx = write code address
562 <1> ;   esi = IOCTL code address
563 <1> ;   edi = dispatch table address
564 <1> ;   ebp = address table address
565 <1> ; Address Table:
566 <1> ;   Offset 0 : open code address
567 <1> ;   Offset 4 : read code address
568 <1> ;   Offset 8 : write code address
569 <1> ;   Offset 12 : close code address
570 <1> ;   Offset 16 : IOCTL code address
571 <1> ;   Offset 20 : initialization code address
572 <1> ;   Offset 24 : description table address
573 <1> ;   Offset 28 : configuration table address
574 <1> ;   Offset 32 : device name address
575 <1> ;   Offset 36 : dispatch table address
576 <1> ;       (for calling kernel-device functions)
577 <1>
578 0001460C <res 00000040> <1> IDEV_NAME: resb 8*NUMIDEV
579 <1> ; 8 byte names of installable device drivers
580 <1>
581 0001464C <res 00000008> <1> IDEV_TYPE: resb NUMIDEV ; Driver type of installable device drivers
582 00014654 <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 0001465C <res 00000020> <1> IDEV_OADDR: resd NUMIDEV ; open function addr for installable dev driver
586 0001467C <res 00000020> <1> IDEV_CADDR: resd NUMIDEV ; close function addr for installable dev driver
587 0001469C <res 00000020> <1> IDEV_RADDR: resd NUMIDEV ; read function addr for installable dev driver
588 000146BC <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 000146DC <res 0000001E> <1> DEV_ACCESS: resb NUMOFDEVICES ; bit 0 = accessible 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 000146FA <res 0000001E> <1> DEV_R_OWNER: resb NUMOFDEVICES ; Reading owner no (u.uid) of devices
602 00014718 <res 0000001E> <1> DEV_R_OPENCOUNT: resb NUMOFDEVICES ; Reading open count
603 00014736 <res 0000001E> <1> DEV_W_OWNER: resb NUMOFDEVICES ; Writing owner no (u.uid) of devices
604 00014754 <res 0000001E> <1> DEV_W_OPENCOUNT: resb NUMOFDEVICES ; Writing open count
605 00014772 <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 00014790 <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 000147AE <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 00014826 <res 00000078> <1> DEV_R_POINTER: resd NUMOFDEVICES ; reading pointer, writing pointer
618 0001489E <res 00000078> <1> DEV_W_POINTER: resd NUMOFDEVICES ; sector number if block device
619 <1> ; character offset if char device
620 00014916 <res 00000002> <1> alignb 4
621 <1>
622 <1> ; 06/10/2016
623 <1> ; Open File Parameters
624 00014918 <res 00000028> <1> OF_FCLUSTER: resd OPENFILES ; First clusters of open files
625 00014940 <res 0000000A> <1> OF_DRIVE: resb OPENFILES ; Logical DOS drive numbers of open files
626 0001494A <res 0000000A> <1> OF_MODE: resb OPENFILES ; Open mode (1 = read, 2 = write, 3 = r&w)
627 00014954 <res 0000000A> <1> OF_STATUS: resb OPENFILES ; (bit 0 = read, bit 1 = write)
628 0001495E <res 0000000A> <1> OF_OPENCOUNT: resb OPENFILES ; Open counts of open files
629 00014968 <res 00000028> <1> OF_POINTER: resd OPENFILES ; File seek/read/write pointer
630 00014990 <res 00000028> <1> OF_SIZE: resd OPENFILES ; File sizes of open files (in bytes)
631 000149B8 <res 00000028> <1> OF_DIRFCLUSTER: resd OPENFILES ; Directory First Clusters of open files
632 000149E0 <res 00000028> <1> OF_DIRCLUSTER: resd OPENFILES ; Directory (Entry) Clusters of open files
633 00014A08 <res 00000028> <1> OF_VOLUMEID: resd OPENFILES ; Vol ID for removable drives of open files
634 00014A30 <res 00000028> <1> OF_OCLUSTER: resd OPENFILES ; Current clusters of open files
635 00014A58 <res 00000028> <1> OF_CCINDEX: resd OPENFILES ; Cluster index numbers of current clusters

```

```

636
637 00014A80 <res 00000014>
638
639
640
641 00014A94 <res 00000060>
642
643
644
645
646 00014AF4 <res 00000001>
647 00014AF5 <res 00000001>
648
649 00014AF6 <res 00000001>
650
651
652
653
654
655 00014AF7 <res 00000001>
656
657
658
659
660
661 00014AF8 <res 00000009>
662 00014B01 <res 00000009>
663 00014B0A <res 00000009>
664 00014B13 <res 00000009>
665 00014B1C <res 00000024>
666
667
668 00014B40 <res 00000004>
669 00014B44 <res 00000001>
2827
2828
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29 00014B45 <res 00000001>
30
31
32
33
34
35
36 00014B46 <res 00000002>
37 00014B48 <res 00000001>
38 00014B49 <res 00000001>
39
40 00014B4A <res 00000002>
41 00014B4C <res 00000010>
42 00014B5C <res 00000004>
43 00014B60 <res 00000004>
44 00014B64 <res 00000002>
45
46
47
48
49
50
51
52
53
54
55
56
57 00014B66 <res 00000020>
58 00014B86 <res 00000020>
59 00014BA6 <res 00000020>
60 00014BC6 <res 00000010>
61 00014BD6 <res 00000010>
62 00014BE6 <res 00000010>
63 00014BF6 <res 00000010>
64
65
66 00014C06 <res 00000040>

<1> ; 24/10/2016
<1> OF_DIRENTRY: resw OPENFILES ; Directory entry index no. in dir cluster
<1> ; Sector index = entry index / 16
<1> ;alignb 2
<1>
<1> DTA: resd 24 ; Find First File data transfer area
<1>
<1> ; 24/02/2017
<1> rcallback: ; IRQ callback method flag for 'syscalbac'
<1> ; 19/12/2016
<1> tcallback: resb 1 ; Timer callback method flag for 'systimer'
<1> trtc: resb 1 ; Timer interrupt type flag for 'systimer'
<1> ; 20/02/2017
<1> no_page_swap: resb 1 ; Swap lock for Signal Response Byte pages
<1> ;15/01/2017
<1> ; 02/01/2017
<1> ;intflg: resb 1 ; software interrupt in progress signal
<1> ; (for timer interrupt)
<1>
<1> alignb 4
<1>
<1> ; 26/02/2017 ; IRQ Callback parameters ('syscalbac')
<1> ;Index: ; 0 to 8
<1> ; 0 = IRQ3, 1 = IRQ4, 2 = IRQ5, 3 = IRQ7
<1> ; 4 = IRQ9, 5 = IRQ10, 6 = IRQ11, 7 = IRQ12, 8 = IRQ13
<1> IRQ.owner: resb 9 ; owner, 0 = free, >0 = [u.uno]
<1> IRQ.dev: resb 9 ; 0 = default/kernel, >0 = device number
<1> IRQ.method: resb 9 ; 0 = Signal Response Byte, 1 = Callback
<1> IRQ.srb: resb 9 ; Signal Response/Return Byte value
<1> IRQ.addr: resd 9 ; Rignal Response Byte address (physical)
<1> ; or Callback service address (virtual)
<1> ; 28/02/2017
<1> IRQ_cr3: resd 1 ; for saving cr3 register in IRQ handler
<1> IRQnum: resb 1 ; IRQ number for IRQ handler (trdosk8.s)
<1> ; 24/01/2016
<1> %include 'ubss.s' ; UNINITIALIZED KERNEL (USER) DATA
<1> ; *****
<1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.0) - UNINITIALIZED USER DATA : ubss.s
<1> ; -----
<1> ; Last Update: 28/02/2017
<1> ; -----
<1> ; Beginning: 24/01/2016
<1> ; -----
<1> ; Assembler: NASM version 2.11 (trdos386.s)
<1> ; -----
<1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
<1> ; ux.s (04/12/2015)
<1> ; *****
<1>
<1> ; Retro UNIX 386 v1 Kernel - ux.s
<1> ; Last Modification: 04/12/2015
<1> ;
<1> ; ////////// RETRO UNIX 386 V1 SYSTEM DEFINITIONS //////////
<1> ; (Modified from
<1> ; Retro UNIX 8086 v1 system definitions in 'UNIX.ASM', 01/09/2014)
<1> ; ((UNIX.ASM (RETRO UNIX 8086 V1 Kernel), 11/03/2013 - 01/09/2014))
<1> ; -----
<1> ; Derived from UNIX Operating System (v1.0 for PDP-11)
<1> ; (Original) Source Code by Ken Thompson (1971-1972)
<1> ; <Bell Laboratories (17/3/1972)>
<1> ; <Preliminary Release of UNIX Implementation Document>
<1> ; (Section E10 (17/3/1972) - ux.s)
<1> ; *****
<1>
<1> alignb 2
<1>
<1> inode:
<1> ; 11/03/2013.
<1> ;Derived from UNIX v1 source code 'inode' structure (ux).
<1> ;i.
<1>
<1> i.flgs: resw 1
<1> i.nlks: resb 1
<1> i.uid: resb 1
<1> ;i.size: resw 1 ; size
<1> resw 1 ; 29/04/2016
<1> i.dskp: resw 8 ; 16 bytes
<1> i.ctim: resd 1
<1> i.mtim: resd 1
<1> i.rsvd: resw 1 ; Reserved (ZERO/Undefined word for UNIX v1.)
<1>
<1> I_SIZE equ $ - inode
<1>
<1> process:
<1> ; 19/12/2016
<1> ; 21/05/2016
<1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
<1> ; 06/05/2015 - Retro UNIX 386 v1
<1> ; 11/03/2013 - 05/02/2014 (Retro UNIX 8086 v1)
<1> ;Derived from UNIX v1 source code 'proc' structure (ux).
<1> ;p.
<1>
<1> p.pid: resw nproc
<1> p.ppid: resw nproc
<1> p.break: resw nproc
<1> p.ttyc: resb nproc ; console tty in Retro UNIX 8086 v1.
<1> p.waitc: resb nproc ; waiting channel in Retro UNIX 8086 v1.
<1> p.link: resb nproc
<1> p.stat: resb nproc
<1>
<1> ; 06/05/2015 (Retro UNIX 386 v1 feature only !)
<1> p.upage: resd nproc ; Physical address of the process's

```



```
67 <1> ; 'user' structure
68 <1> ; 21/05/2016
69 <1> ; 19/05/2016 (TRDOS 386 feature only!)
70 00014C46 <res 00000010> <1> p.timer: resb nproc ; number of timer events of the processs
71 <1>
72 <1> ; 19/12/2016
73 00014C56 <res 00000040> <1> p.tcb: resd nproc ; timer callback service address (if > 0)
74 <1>
75 <1> P_SIZE equ $ - process
76 <1>
77 <1> ; fsp table (original UNIX v1)
78 <1> ;
79 <1> ;Entry
80 <1> ; 15 0
81 <1> ; 1 |---|-----|
82 <1> ; |r/w| i-number of open file
83 <1> ; |---|-----|
84 <1> ; | device number
85 <1> ; |-----|
86 <1> ; (*) | offset pointer, i.e., r/w pointer to file
87 <1> ; |-----|
88 <1> ; | flag that says | number of processes
89 <1> ; | file deleted | that have file open
90 <1> ; |-----|
91 <1> ; 2
92 <1> ; |-----|
93 <1> ; |-----|
94 <1> ; |-----|
95 <1> ; |-----|
96 <1> ; |-----|
97 <1> ; |-----|
98 <1> ; |-----|
99 <1> ; 3
100 <1> ;
101 <1> ;
102 <1> ; (*) Retro UNIX 386 v1 modification: 32 bit offset pointer
103 <1>
104 <1>
105 <1> ; 15/04/2015
106 00014C96 <res 000001F4> <1> fsp: resb nfiles * 10 ; 11/05/2015 (8 -> 10)
107 00014E8A <res 00000002> <1> idev: resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
108 00014E8C <res 00000002> <1> cdev: resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
109 <1> ; 18/05/2015
110 <1> ; 26/04/2013 device/drive parameters (Retro UNIX 8086 v1 feature only!)
111 <1> ; 'UNIX' device numbers (as in 'cdev' and 'u.cdrv')
112 <1> ; 0 -> root device (which has Retro UNIX 8086 v1 file system)
113 <1> ; 1 -> mounted device (which has Retro UNIX 8086 v1 file system)
114 <1> ; 'Retro UNIX 8086 v1' device numbers: (for disk I/O procedures)
115 <1> ; 0 -> fd0 (physical drive, floppy disk 1), physical drive number = 0
116 <1> ; 1 -> fd1 (physical drive, floppy disk 2), physical drive number = 1
117 <1> ; 2 -> hd0 (physical drive, hard disk 1), physical drive number = 80h
118 <1> ; 3 -> hd1 (physical drive, hard disk 2), physical drive number = 81h
119 <1> ; 4 -> hd2 (physical drive, hard disk 3), physical drive number = 82h
120 <1> ; 5 -> hd3 (physical drive, hard disk 4), physical drive number = 83h
121 00014E8E <res 00000001> <1> rdev: resb 1 ; root device number ; Retro UNIX 8086 v1 feature only!
122 <1> ; as above, for physical drives numbers in following table
123 00014E8F <res 00000001> <1> mdev: resb 1 ; mounted device number ; Retro UNIX 8086 v1 feature only!
124 <1> ; 15/04/2015
125 00014E90 <res 00000001> <1> active: resb 1
126 00014E91 <res 00000001> <1> resb 1 ; 09/06/2015
127 00014E92 <res 00000002> <1> mnti: resw 1
128 00014E94 <res 00000002> <1> mpid: resw 1
129 00014E96 <res 00000002> <1> rootdir: resw 1
130 <1>
131 <1> ; 21/05/2016 - TRDOS 386 (TRDOS v2.0) - priority levels, 3 run queues
132 <1> runq:
133 00014E98 <res 00000002> <1> runq_event: resw 1 ; high priority, 'run for event' ; 2
134 00014E9A <res 00000002> <1> runq_normal: resw 1 ; normal/regular priority, 'run as regular' ; 1
135 00014E9C <res 00000002> <1> runq_background: resw 1 ; low priority, 'run on background' ; 0
136 <1> ;
137 00014E9E <res 00000001> <1> imod: resb 1
138 00014E9F <res 00000001> <1> smod: resb 1
139 00014EA0 <res 00000001> <1> mmmod: resb 1
140 00014EA1 <res 00000001> <1> sysflg: resb 1
141 <1>
142 00014EA2 <res 00000002> <1> alignb 4
143 <1>
144 <1> user:
145 <1> ; 13/01/2017
146 <1> ; 19/12/2016
147 <1> ; 21/05/2016 - TRDOS 386 (TRDOS v2.0)
148 <1> ; [u.pri] usage method modification
149 <1> ; 04/12/2015
150 <1> ; 18/10/2015
151 <1> ; 12/10/2015
152 <1> ; 21/09/2015
153 <1> ; 24/07/2015
154 <1> ; 16/06/2015
155 <1> ; 09/06/2015
156 <1> ; 11/05/2015
157 <1> ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit modifications)
158 <1> ; 10/10/2013
159 <1> ; 11/03/2013.
160 <1> ;Derived from UNIX v1 source code 'user' structure (ux).
161 <1> ;u.
162 <1>
163 00014EA4 <res 00000004> <1> u.sp: resd 1 ; esp (kernel stack at the beginning of 'sysent')
164 00014EA8 <res 00000004> <1> u.usp: resd 1 ; esp (kernel stack points to user's registers)
165 00014EAC <res 00000004> <1> u.r0: resd 1 ; eax
166 00014EB0 <res 00000002> <1> u.cdir: resw 1
167 00014EB2 <res 0000000A> <1> u.fp: resb 10
168 00014EBC <res 00000004> <1> u.fofp: resd 1
```

```

169 00014EC0 <res 00000004> <1> u.dirp: resd 1
170 00014EC4 <res 00000004> <1> u.namep: resd 1
171 00014EC8 <res 00000004> <1> u.off: resd 1
172 00014ECC <res 00000004> <1> u.base: resd 1
173 00014ED0 <res 00000004> <1> u.count: resd 1
174 00014ED4 <res 00000004> <1> u.nread: resd 1
175 00014ED8 <res 00000004> <1> u.break: resd 1 ; break
176 00014EDC <res 00000002> <1> u.ttyp: resw 1
177 <1> ; 10/01/2017 (TRDOS 386, relocation and dword alignment)
178 <1> ; tty number (rtty, rcvt, wtty)
179 00014EDE <res 00000001> <1> u.ttyn: resb 1 ; 28/07/2013 - Retro Unix 8086 v1 feature only !
180 00014EDF <res 00000001> <1> u.resb: resb 1 ; 10/01/2017 (TRDOS 386, temporary)
181 00014EE0 <res 00000010> <1> u.dirbuf: resb 16 ; 04/12/2015 (10 -> 16)
182 <1> ;u.pri: resw 1 ; 14/02/2014
183 00014EF0 <res 00000001> <1> u.quant: resb 1 ; Retro UNIX 8086 v1 Feature only ! (uquant)
184 00014EF1 <res 00000001> <1> u.pri: resb 1 ; Modification: 21/05/2016 (priority levels: 0, 1, 2)
185 00014EF2 <res 00000002> <1> u.intr: resw 1
186 00014EF4 <res 00000002> <1> u.quit: resw 1
187 <1> ;u.emt: resw 1 ; 10/10/2013
188 <1> ;u.ilgins: resw 1 ; 10/01/2017
189 00014EF6 <res 00000002> <1> u.cdrv: resw 1 ; cdev
190 00014EF8 <res 00000001> <1> u.uid: resb 1 ; uid
191 00014EF9 <res 00000001> <1> u.ruid: resb 1
192 00014EFA <res 00000001> <1> u.bsys: resb 1
193 00014EFB <res 00000001> <1> u.uno: resb 1
194 00014EFC <res 00000004> <1> u.upage: resd 1 ; 16/04/2015 - Retro Unix 386 v1 feature only !
195 00014F00 <res 00000004> <1> u.pgdir: resd 1 ; 09/03/2015 (page dir addr of process)
196 00014F04 <res 00000004> <1> u.ppgdir: resd 1 ; 06/05/2015 (page dir addr of the parent process)
197 00014F08 <res 00000004> <1> u.pbase: resd 1 ; 20/05/2015 (physical base/transfer address)
198 00014F0C <res 00000002> <1> u.pcount: resw 1 ; 20/05/2015 (byte -transfer- count for page)
199 <1> ;u.pncount: resw 1
200 <1> ; 16/06/2015 (byte -transfer- count for page, 'namei', 'mkdir')
201 <1> ;u.pnbase: resd 1
202 <1> ; 16/06/2015 (physical base/transfer address, 'namei', 'mkdir')
203 <1> ; 09/06/2015
204 00014F0E <res 00000001> <1> u.kcall: resb 1 ; The caller is 'namei' (dskr) or 'mkdir' (dskw) sign
205 00014F0F <res 00000001> <1> u.brwdev: resb 1 ; Block device number for direct I/O (bread & bwrite)
206 <1> ; 24/07/2015 - 24/06/2015
207 <1> ;u.args: resd 1 ; arguments list (line) offset from start of [u.upage]
208 <1> ; (arg list/line is from offset [u.args] to 4096 in [u.upage])
209 <1> ; ([u.args] points to argument count -argc- address offset)
210 <1> ; 24/06/2015
211 <1> ;u.core: resd 1 ; physical start address of user's memory space (for sys exec)
212 <1> ;u.ecore: resd 1 ; physical end address of user's memory space (for sys exec)
213 <1> ; last error number
214 00014F10 <res 00000004> <1> u.error: resd 1 ; 28/07/2013 - 09/03/2015
215 <1> ; Retro UNIX 8086/386 v1 feature only!
216 <1> ; 21/09/2015 (debugging - page fault analyze)
217 00014F14 <res 00000004> <1> u.pfcount: resd 1 ; page fault count for (this) process (for sys geterr)
218 <1> ; 19/12/2016 (TRDOS 386)
219 00014F18 <res 00000004> <1> u.tcb: resd 1 ; Timer callback address/flag which will be used by timer int
220 <1> ; 13/01/2017 (TRDOS 386)
221 00014F1C <res 00000001> <1> u.t_lock: resb 1 ; Timer interrupt (callback) lock (unlocked by 'sysrele')
222 00014F1D <res 00000001> <1> u.t_mode: resb 1 ; running mode during timer interrupt (0= system, 0FFh= user)
223 <1> ; 26/02/2017 (TRDOS 386)
224 00014F1E <res 00000001> <1> u.irqc: resb 1 ; Count of IRQ callback services (IRQs in use)
225 <1> ; 28/02/2017 (TRDOS 386)
226 00014F1F <res 00000001> <1> u.irqwait: resb 1 ; IRQ waiting for callback service flag (IRQ number, If > 0)
227 00014F20 <res 00000001> <1> u.r_lock: resb 1 ; 'IRQ callback service is in progress' flag (IRQ lock)
228 00014F21 <res 00000001> <1> u.r_mode: resb 1 ; running mode during hardware interrupt
229 <1> ; 27/02/2017 (TRDOS 386)
230 00014F22 <res 00000001> <1> u.fpsave: resb 1 ; TRDOS 386, 'save/restore FPU registers' flag
231 00014F23 <res 00000001> <1> alignb 4
232 00014F24 <res 0000005E> <1> u.fpregs: resb 94 ; 94 byte area for saving and restoring FPU registers
233 <1>
234 00014F82 <res 00000002> <1> alignb 4
235 <1>
236 <1> U_SIZE equ $ - user
237 <1>
238 <1> ; 18/10/2015 - Retro UNIX 386 v1 (local variables for 'namei' and 'sysexec')
239 00014F84 <res 00000004> <1> pcore: resd 1 ; physical start address of user's memory space (for sys exec)
240 00014F88 <res 00000004> <1> ecore: resd 1 ; physical start address of user's memory space (for sys exec)
241 00014F8C <res 00000004> <1> nbase: resd 1 ; physical base address for 'namei' & 'sysexec'
242 00014F90 <res 00000002> <1> ncount: resw 1 ; remain byte count in page for 'namei' & 'sysexec'
243 00014F92 <res 00000002> <1> argc: resw 1 ; argument count for 'sysexec'
244 00014F94 <res 00000004> <1> argv: resd 1 ; argument list (recent) address for 'sysexec'
245 <1>
246 <1> ; 03/06/2015 - Retro UNIX 386 v1 Beginning
247 <1> ; 07/04/2013 - 31/07/2013 - Retro UNIX 8086 v1
248 00014F98 <res 00000001> <1> rw: resb 1 ; Read/Write sign (iget)
249 <1>
250 <1> ;alignb 4
251 <1>
252 <1> ; 24/04/2016
253 00014F99 <res 00000004> <1> ii: resd 1 ; first cluster of the program file
254 00014F9D <res 00000004> <1> i.size: resd 1 ; size of the program file
2829
2830 00014FA1 <res 00000003> alignb 4
2831
2832 ; 23/05/2016 (TRDOS 386)
2833 ; 14/10/2015 (Retro UNIX 386 v1, 'unix386.s')
2834 00014FA4 <res 00000004> cr3reg: resd 1 ; cr3 register content at the beginning of the timer
2835 ; (or RTC) interrupt handler.
2836
2837 ; 10/12/2016 (callback)
2838 ; 10/06/2016
2839 ; 19/05/2016
2840 ; 18/05/2016 - TRDOS 386 feature only !
2841 00014FA8 <res 00000100> timer_set: resd 16*4 ; 256 bytes memory space for 16 timer events
2842 ; Timer Event Structure: (max. 16 timer events, 16*16 bytes)
2843 ; Owner: resb 1 ; 0 = free
2844 ; ;>0 = process number (u.uno)

```

```

2845             ;      Callback:      resb 1 ; 0 = response byte address (phy)
2846             ;                      1 = callback address (virtual)
2847             ;      Interrupt:      resb 1 ; 0 = Timer interrupt (or none)
2848             ;                      ; 1 = Real Time Clock interrupt
2849             ;      Response:      resb 1 ; 0 to 255, signal return value
2850             ;      Count Limit: resd 1 ; count of ticks (total/set)
2851             ;      Current Count:  resd 1 ; count of ticks (current)
2852             ;      Response Addr:  resd 1 ; response byte (pointer) address
2853             ;                      ; (or callback -user service- address)
2854
2855             ;; Memory (swap) Data (11/03/2015)
2856             ; 09/03/2015
2857 000150A8 <res 00000002> swpq_count: resw 1 ; count of pages on the swap queue
2858 000150AA <res 00000004> swp_drv:   resd 1 ; logical drive description table address of the swap drive/disk
2859 000150AE <res 00000004> swpd_size: resd 1 ; size of swap drive/disk (volume) in sectors (512 bytes).

2860 000150B2 <res 00000004> swpd_free: resd 1 ; free page blocks (4096 bytes) on swap disk/drive (logical)
2861 000150B6 <res 00000004> swpd_next: resd 1 ; next free page block
2862 000150BA <res 00000004> swpd_last: resd 1 ; last swap page block
2863
2864 000150BE <res 00000002> alignb 4
2865
2866             ; 10/07/2015
2867             ; 28/08/2014
2868 000150C0 <res 00000004> error_code: resd 1
2869             ; 29/08/2014
2870 000150C4 <res 00000004> FaultOffset:      resd 1
2871             ; 21/09/2015
2872 000150C8 <res 00000004> PF_Count:   resd 1 ; total page fault count
2873             ; (for debugging - page fault analyze)
2874             ; 'page_fault_handler' (memory.inc)
2875             ; 'sysgeterr' (u9.s)
2876
2877             ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
2878             ; 22/08/2015 (Retro UNIX 386 v1)
2879             buffer:
2880             resb 8
2881             readi_buffer:
2882             resb 512
2883 000152D4 <res 00000008> resb 8
2884             writei_buffer:
2885             resb 512
2886             ; 24/10/2016
2887 000154DC <res 00000008> resb 8
2888             rw_buffer:
2889 000154E4 <res 00000800> resb 2048 ; general purposed, r/w sector buffer
2890
2891             bss_end:
2892
2893             ; 27/12/2013
2894             _end: ; end of kernel code

```