

```

1  ; *****
2  ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.10
3  ;
4  ; Last update: 10/01/2026 (Previous: 28/12/2025)
5  ;
6  ; Beginning: 04/01/2016
7  ;
8  ; Assembler: NASM version 2.15 (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 ; nasm trdos386.s -l trdos386.txt -o TRDOS386.SYS
22 ;
23 KLOAD      equ 10000h ; kernel loading address
24 ; NOTE: Retro UNIX 8086 v1 boot code loads kernel at 1000h:0000h
25 KCODE      equ 08h ; Code segment descriptor (ring 0)
26 KDATA      equ 10h ; Data segment descriptor (ring 0)
27 ; 19/03/2015
28 UCODE      equ 18h ; 18h + 3h (ring 3)
29 UDATA      equ 23h ; 20h + 3h (ring 3)
30 ; 24/03/2015
31 TSS equ 28h ; Task state segment descriptor (ring 0)
32 ; 19/03/2015
33 CORE equ 400000h ; Start of USER's virtual/linear address space
34 ; (at the end of the 1st 4MB)
35 ECORE      equ 0FFC0000h ; End of USER's virtual address space (4GB - 4MB)
36 ; ULIMIT = (ECORE/4096) - 1 = 0FFBFFh (in GDT)
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 ; IBM PC/AT BIOS ---- 10/06/85 (postequ.inc)
43 ;----- CMOS TABLE LOCATION ADDRESS'S -----
44 CMOS_SECONDS EQU 00H ; SECONDS (BCD)
45 CMOS_SEC_ALARM EQU 01H ; SECONDS ALARM (BCD)
46 CMOS_MINUTES EQU 02H ; MINUTES (BCD)
47 CMOS_MIN_ALARM EQU 03H ; MINUTES ALARM (BCD)
48 CMOS_HOURS EQU 04H ; HOURS (BCD)
49 CMOS_HR_ALARM EQU 05H ; HOURS ALARM (BCD)
50 CMOS_DAY_WEEK EQU 06H ; DAY OF THE WEEK (BCD)
51 CMOS_DAY_MONTH EQU 07H ; DAY OF THE MONTH (BCD)
52 CMOS_MONTH EQU 08H ; MONTH (BCD)
53 CMOS_YEAR EQU 09H ; YEAR (TWO DIGITS) (BCD)
54 CMOS_CENTURY EQU 32H ; DATE CENTURY BYTE (BCD)
55 CMOS_REG_A EQU 0AH ; STATUS REGISTER A
56 CMOS_REG_B EQU 0BH ; STATUS REGISTER B ALARM
57 CMOS_REG_C EQU 0CH ; STATUS REGISTER C FLAGS
58 CMOS_REG_D EQU 0DH ; STATUS REGISTER D BATTERY
59 CMOS_SHUT_DOWN EQU 0FH ; SHUTDOWN STATUS COMMAND BYTE
60 ;
61 ; CMOS EQUATES FOR THIS SYSTEM ;
62 ;-----
63 CMOS_PORT EQU 070H ; I/O ADDRESS OF CMOS ADDRESS PORT
64 CMOS_DATA EQU 071H ; I/O ADDRESS OF CMOS DATA PORT
65 NMI EQU 10000000B ; DISABLE NMI INTERRUPTS MASK -
66 ; HIGH BIT OF CMOS LOCATION ADDRESS
67 ;
68 ; Memory Allocation Table Address
69 ; 05/11/2014
70 ; 31/10/2014
71 MEM_ALLOC_TBL equ 100000h ; Memory Allocation Table at the end of
72 ; the 1st 1 MB memory space.
73 ; (This address must be aligned
74 ; on 128 KB boundary, if it will be
75 ; changed later.)
76 ; ((lower 17 bits of 32 bit M.A.T.
77 ; address must be ZERO)).
78 ; (((Reason: 32 bit allocation
79 ; instructions, dword steps)))
80 ; (((byte >> 12 --> page >> 5)))
81 ; 04/11/2014
82 PDE_A_PRESENT equ 1 ; Present flag for PDE
83 PDE_A_WRITE equ 2 ; writable (write permission) flag
84 PDE_A_USER equ 4 ; User (non-system/kernel) page flag
85 ;
86 PTE_A_PRESENT equ 1 ; Present flag for PTE (bit 0)
87 PTE_A_WRITE equ 2 ; writable (write permission) flag (bit 1)
88 PTE_A_USER equ 4 ; User (non-system/kernel) page flag (bit 2)
89 PTE_A_ACCESS equ 32 ; Accessed flag (bit 5) ; 09/03/2015
90 ;
91 ; 17/02/2015 (unix386.s)
92 ; 10/12/2014 - 30/12/2014 (0B000h -> 9000h) (dsectrm2.s)
93 DPT_SEGM equ 09000h ; FDPT segment (EDD v1.1, EDD v3)
94 ;
95 HD0_DPT equ 0 ; Disk parameter table address for hd0
96 HD1_DPT equ 32 ; Disk parameter table address for hd1
97 HD2_DPT equ 64 ; Disk parameter table address for hd2
98 HD3_DPT equ 96 ; Disk parameter table address for hd3
99 ;
100 ; 15/11/2020
101 VBE3INFOSEG equ 97E0h ; 512 bytes before Video_Pg_Backup
102 ; 15/12/2020
103 VBE3MODEINFOSEG equ 97C0h ; 512 bytes before VBE3INFOBLOCK
104 ;
105 ; 29/11/2020
106 VBE3INFOBLOCK equ 97E00h ; linear address (512 bytes)
107 VBE3MODEINFOBLOCK equ 97C00h ; linear address (256 bytes)
108 VBE3SAVERSTOREBLOCK equ 97600h ; linear address (2048 bytes)
109 VBE3CRTINFOBLOCK equ 97D80h ; linear address (64 bytes) ; 17/01/2021
110 VBE3BIOSDATABLOCK equ 97000h ; linear address (1536 bytes)
111 VBE3STACKADDR equ 96000h ; linear address (1024 bytes)
112 ; VBE3 32 bit Protected Mode Interface (16 bit) Selectors (in GDT)
113 VBE3CS equ 30h ; _vbe3_CS:
114 VBE3BDS equ 38h ; _vbe3_BDS:
115 VBE3A000 equ 40h ; _A0000Sel:
116 VBE3B000 equ 48h ; _B0000Sel:
117 VBE3B800 equ 50h ; _B8000Sel:
118 VBE3DS equ 58h ; _vbe3_DS:
119 VBE3SS equ 60h ; _vbe3_SS:
120 VBE3ES equ 68h ; _vbe3_ES:
121 KCODE16 equ 70h ; _16bit_CS:
122 ; 14/01/2021
123 ; 06/12/2020
124 VBE3VIDEOSTATE equ 95800h ; 2048 bytes

```

```

125 ; 05/01/2021
126 VGA_FONT16USER equ 94000h ; 8x16 pixels user font (256 chars)
127 ; (reserved/allocated font space: 4096 bytes)
128
129 VGA_FONT8USER equ 95000h ; 8x8 pixels user font (256 chars)
130 ; (reserved/allocated font space: 2048 bytes)
131
132 ; 17/01/2021
133 ; temporary (initial) location for EDID information
134 VBE3EDIDINFOBLOCK equ 97D00h ; linear address (128 bytes)
135
136 ; FDPT (Phoenix, Enhanced Disk Drive Specification v1.1, v3.0)
137 ; (HDPT: Programmer's Guide to the AMIBIOS, 1993)
138
139 FDPT_CYLS equ 0 ; 1 word, number of cylinders
140 FDPT_HDS equ 2 ; 1 byte, number of heads
141 FDPT_TT equ 3 ; 1 byte, A0h = translated FDPT with logical values
142 ; otherwise it is standard FDPT with physical values
143 FDPT_PCOMP equ 5 ; 1 word, starting write precompensation cylinder
144 ; (obsolete for IDE/ATA drives)
145 FDPT_CB equ 8 ; 1 byte, drive control byte
146 ; Bits 7-6 : Enable or disable retries (00h = enable)
147 ; Bit 5: 1 = Defect map is located at last cyl. + 1
148 ; Bit 4 : Reserved. Always 0
149 ; Bit 3 : Set to 1 if more than 8 heads
150 ; Bit 2-0 : Reserved. Always 0
151 FDPT_LZ equ 12 ; 1 word, landing zone (obsolete for IDE/ATA drives)
152 FDPT_SPT equ 14 ; 1 byte, sectors per track
153
154 ; Floppy Drive Parameters Table (Programmer's Guide to the AMIBIOS, 1993)
155 ; (11 bytes long) will be used by diskette handler/bios
156 ; which is derived from IBM PC-AT BIOS (DISKETTE.ASM, 21/04/1986).
157
158 ; 01/02/2016
159 LogicalDOSDisks equ 90000h + 100h ; 26*256 = 6656 bytes
160 Directory_Buffer equ 80000h ; max = 64K Bytes
161 FAT_Buffer equ 91C00h ; 1536 bytes (3 sectors)
162 ; 15/02/2016
163 Cluster_Buffer equ 70000h ; max = 64K Bytes ; buffer for file read & write
164 ; 11/04/2016
165 Env_Page equ 93000h ; 512 bytes (4096 bytes)
166 Env_Page_Size equ 512 ; (4096 bytes)
167 ; 30/07/2016
168 Video_Pg_Backup equ 98000h ; Mode 3h, video page backup (32K, 8 pages)
169
170 ; 29/11/2020
171 ; Free/Reserved memory blocks (in 1st 1MB): 93200h to 96000h (available)
172 ; 06/12/2020
173 ; Free/Reserved memory blocks (in 1st 1MB): 93200h to 95800h (available)
174
175 ; 15/12/2020
176 LFB_ADDR equ LFB_Info+LFBINFO.LFB_addr
177 LFB_SIZE equ LFB_Info+LFBINFO.LFB_size
178
179 ; 04/12/2023 - TRDOS 386 v2.0.7
180 SYSTEMSTACK_ADDR equ 97000h ; max. 3072 bytes (96400h-97000h)
181 VBE3BIOSCODE_ADDR equ 60000h ; Protected Mode Video Bios (64KB)
182 ; AC97DMABUFFER_ADDR equ 40000h ; AC97 & VIA VT8233 DMA Buffer (128KB)
183 SB16DMABUFFER_ADDR equ 50000h ; Sound Blaster 16 DMA Buffer (64KB)
184 sb16_dma_buffer equ SB16DMABUFFER_ADDR
185
186 ; 29/08/2023 - TRDOS 386 v2.0.6
187 ; 30/11/2020
188 ; 29/11/2020 - TRDOS 386 v2.0.3
189
190 struct PMInfo ; VESA VBE3 PMInfoBlock ('PMID' block)
191 00000000 ???????? .Signature: resb 4 ; db 'PMID' ; PM Info Block Signature
192 00000004 ???? .EntryPoint: resw 1 ; Offset of PM entry point within BIOS
193 00000006 ???? .PMInitialize: resw 1 ; Offset of PM initialization entry point
194 00000008 ???? .BIOSDataSel: resw 1 ; Selector to BIOS data area emulation block
195 0000000A ???? .A0000Sel: resw 1 ; Selector to access A0000h physical mem
196 0000000C ???? .B0000Sel: resw 1 ; Selector to access B0000h physical mem
197 0000000E ???? .B8000Sel: resw 1 ; Selector to access B8000h physical mem
198 00000010 ???? .CodeSegSel: resw 1 ; Selector to access code segment as data
199 00000012 ?? .InProtectMode: resb 1 ; Set to 1 when in protected mode
200 00000013 ?? .Checksum: resb 1 ; Checksum byte for structure
201 .size:
202
203 endstruc
204
205 [BITS 16] ; we need 16-bit instructions for Real mode
206
207 [ORG 0]
208 ; 12/11/2014
209 ; Save boot drive number (that is default root drive)
210 00000000 8816[7866] mov [boot_drv], dl ; physical drv number
211
212 ; Determine installed memory
213 ; 31/10/2014
214 ;
215 00000004 B801E8 mov ax, 0E801h ; Get memory size
216 00000007 CD15 int 15h ; for large configurations
217 00000009 7308 jnc short chk_ms
218 0000000B B488 mov ah, 88h ; Get extended memory size
219 0000000D CD15 int 15h
220 ;
221 ; mov al, 17h ; Extended memory (1K blocks) low byte
222 ; out 70h, al ; select CMOS register
223 ; in al, 71h ; read data (1 byte)
224 ; mov cl, al
225 ; mov al, 18h ; Extended memory (1K blocks) high byte
226 ; out 70h, al ; select CMOS register
227 ; in al, 71h ; read data (1 byte)
228 ; mov ch, al
229 ;
230 0000000F 89C1 mov cx, ax
231 00000011 31D2 xor dx, dx
232 chk_ms:
233 00000013 890E[7466] mov [mem_1m_1k], cx
234 00000017 8916[7666] mov [mem_16m_64k], dx
235 ; 24/11/2023
236 0000001B 8916[180F] mov [real_mem_16m_64k], dx
237
238 ; 05/11/2014
239 ; and dx, dx
240 ; jz short L2
241 0000001F 81F90004 cmp cx, 1024
242 ; jnb short L0
243 00000023 7351 jnb short V0 ; 14/11/2020
244 ; insufficient memory_error
245 ; Minimum 2 MB memory is needed...
246 ; 05/11/2014
247 ; (real mode error printing)
248 00000025 FB sti

```

```

249 00000026 BE[3A00]      mov     si, msg_out_of_memory
250 00000029 BB0700      mov     bx, 7
251 0000002C B40E      mov     ah, 0Eh; write tty
252
253 0000002E AC      oom_1: lodsb
254 0000002F 08C0      or      al, al
255 00000031 7404      jz      short oom_2
256 00000033 CD10      int     10h
257 00000035 EBF7      jmp     short oom_1
258
259 00000037 F4      oom_2: hlt
260 00000038 EBF0      jmp     short oom_2
261
262
263
264
265 0000003A 070D0A      msg_out_of_memory: db 07h, 0Dh, 0Ah
266 0000003D 496E73756666696369- db  'Insufficient memory !'
266 00000046 656E74206D656D6F72-
266 0000004F 792021
267 00000052 0D0A      db 0Dh, 0Ah
268
269 00000054 284D696E696D756D20- _int13h_48h_buffer: ; 07/07/2016
269 0000005D 324D42206D656D6F72- db  '(Minimum 2MB memory is needed.)'
269 00000066 79206973206E656564-
269 0000006F 65642E29
270 00000073 0D0A00      db 0Dh, 0Ah, 0
271
272
273
274
275 00000076 B80300      v0:
276 00000079 CD10      ; 18/10/2023 - TRDOS 386 v2.0.7
277      ; set video mode to 03h again
278      ; (to reset video bios data in ROMBIOS DATA AREA)
278 0000007B BF0097      mov     ax, 3
279 0000007E 8EC7      int     10h
280 00000080 31F6      ; copy IVT and ROMBIOS DATA AREA to VBE3 BIOS data area
281 00000082 31FF      mov     di, VBE3BIOSDATABLOCK>>4
282 00000084 8EDE      mov     es, di
283 00000086 B90003      xor     si, si
284 00000089 F3A5      xor     di, di
285 0000008B 0E      mov     ds, si ; 0
286 0000008C 1F      mov     cx, 300h ; 600h / 2
287      rep movsw
288      push cs
289      pop  ds
290
291      ; 24/11/2023
292      ; 15/12/2020
293      ;mov     si, [mem_16m_64k]
294      ;mov     [real_mem_16m_64k], si
295
296      ; 15/11/2020
297      ; 14/11/2020 (TRDOS 386 v2.0.3)
298      ; check VESA (VBE) VIDEO BIOS version
297 0000008D B8034F      mov     ax, 4F03h ; Return current VBE mode
298 00000090 CD10      int     10h
299 00000092 83F84F      cmp     ax, 004Fh ; successful (vbe) function call
300      jne     short L0 ; not a VESA VBE compatible bios
301      ; 18/10/2023
302 00000095 7561      jne     short v1 ; restore es
303
304      ; 27/11/2023
305      ; 24/11/2023 - temporary
306      jmp     short v1
307
308      ;mov     ah, 3
309      ;;jmp     short v1
310
311      ; 15/11/2020
312 00000097 BBE097      mov     bx, VBE3INFOSEG ; 97E0h for current version
313 0000009A 8EC3      mov     es, bx
314 0000009C 31FF      xor     di, di
315 0000009E 2666C70556424532      mov     dword [es:di], 'VBE2' ; request VESA VBE3 info
316      ; es:di = buffer address (512 bytes)
317      ;mov     ax, 4F00h ; Return VBE controller information
318 000000A6 86E0      xchg    al, ah
319 000000A8 CD10      int     10h
320
321      ; dx = cs
322      ; es = VBE3INFOSEG (97E0h)
323      ; di = 0
324      ; ss = (endofkernelfile/16)+16
325      ; sp = 0FFFEh
326
327 000000AA 83F84F      cmp     ax, 004Fh
328 000000AD 7549      jne     short v1 ; old vga bios (not VESA compatible)
329
330      ; 15/11/2020
331 000000AF 2666813D56455341      cmp     dword [es:di], 'VESA'
332 000000B7 753F      jne     short v1
333
334      ;mov     ax, [es:di+4]
335      ; ax = vbe version in BCD format (0200h or 0300h)
336      ;mov     [vbe3], ah ; version number (major)
337
338      ; 15/11/2020
339 000000B9 268A4505      mov     al, [es:di+5]
340      ; al = high byte of VBE version number (02h or 03h)
341
342 000000BD A2[8609]      mov     [vbe3], al ; version number (major)
343      ; 02h or 03h is expected
344
345      ; 17/01/2021
346 000000C0 B301      ; Read EDID
347 000000C2 31C9      mov     bl, 01h; Read EDID
348      xor     cx, cx ; Controller unit number
349      ; (00 = primary controller)
350 000000C4 31D2      xor     dx, dx ; EDID block number = 0
351 000000C6 B8C097      mov     ax, VBE3MODEINFOSEG ; 97C0h for current version
352 000000CB BF0001      mov     es, ax
353      mov     di, VBE3EDIDINFOBLOCK - VBE3MODEINFOBLOCK
354      ; es:di = temporary address of 128 bytes EDID information
355 000000CE B8154F      mov     ax, 4F15h ; VBE/DDC Services
356 000000D1 CD10      int     10h
357      ;cmp     ax, 4Fh
358      ;jne     short v2
359 000000D3 A2[1942]      ;mov     [edid], al ; 4Fh > 0
360
361      ;v2:
362      ; 17/01/2021
363      xor     di, di
364      ; 15/12/2020
365      ; Get linear frame buffer info (for VESA VBE mode 118h)
366      ;mov     si, VBE3MODEINFOSEG ; 97C0h for current version
367      ;mov     es, si
368      ; di = 0
369      mov     cx, 04118h ; 1024*768, 24 bpp, LFB

```

```

368 000000DB B8014F      mov     ax, 4F01h ; Return VBE mode information
369 000000DE CD10        int     10h
370                      ;cmp     ax, 4Fh
371                      ;jne     short v1
372                      ; 19/12/2020
373                      ;mov     si, [es:di+MODEINFO.PhysBasePtr+2]
374                      ;hw of LFB base address
375                      ; MODEINFO structure starts from offset -2
376 000000E0 268B752A     mov     si, [es:di+MODEINFO.PhysBasePtr] ; hw of LFB addr
377 000000E4 8936[1A0F]   mov     [def_LFB_addr], si ; k_LFB_size = 3145728 bytes
378 000000E8 81EE0001     sub     si, 256
379
380                      ; 15/12/2020
381                      ; check memory and decrease it to 3.5 GB if it is 4GB
382                      ; (reserve upper memory for LFB)
383 000000EC 8B3E[7666]   mov     di, [mem_16m_64k]
384                      ; 24/11/2023
385                      ;mov     [real_mem_16m_64k], di
386
387 000000F0 39F7        cmp     di, si
388 000000F2 7604        jna     short v1
389
390 000000F4 8936[7666]   mov     [mem_16m_64k], si
391
392                      ; VESA VBE3 video hardware
393                      ; (example: NVIDIA GEFORCE FX550, 256 MB)
394                      ; uses upper memory from 0D0000000h to 0DFFFFFFFh
395
396                      ;cmp     di, 0CF00h ; 3328 MB - 16MB
397                      ;jna     short v1 ; <= 3328 MB memory, it is not required
398                      ; decrease
399                      ;cmp     al, 3
400                      ;jb     short v2
401                      ; VESA VBE 3
402                      ;mov     word [mem_16m_64k], 0CF00h ; 3328 MB - 16MB
403                      ;jmp     short v1
404
;V2:
405                      ; VESA VBE 2
406                      ; Check Bochs/Qemu/VirtualBox Emulator
407                      ; LFB base address: 0E0000000h
408                      ;sub     ax, ax ; 0
409                      ;mov     dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
410                      ;out     dx, ax ; VBE_DISPI_INDEX_ID register
411                      ;inc     dx
412                      ;in     ax, dx
413                      ;and     al, 0F0h
414                      ;cmp     ax, 0B0C0h
415                      ;jne     short v1
416                      ;
417                      ; BOCHS/QEMU/VIRTUALBOX
418                      ;mov     word [mem_16m_64k], 0DF00h ; 3584 MB - 16MB
419
v1:
420 000000F8 1E          push     ds
421 000000F9 07          pop      es ; restore extra data segment
422
L0:
423
424 %include 'diskinit.s' ; 07/03/2015
425
<1> ; *****
426
<1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.5 - diskinit.s
427
<1> ; -----
428
<1> ; Last Update: 09/08/2022 (Previous: 29/08/2020 - Kernel v2.0.4)
429
<1> ; -----
430
<1> ; Beginning: 24/01/2016
431
<1> ; -----
432
<1> ; Assembler: NASM version 2.15 (trdos386.s)
433
<1> ; -----
434
<1> ; Turkish Rational DOS
435
<1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
436
<1> ;
437
<1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
438
<1> ; diskinit.inc (10/07/2015)
439
<1> ;
440
<1> ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
441
<1> ; *****
442
<1> ;
443
<1> ; Retro UNIX 386 v1 Kernel - DISKINIT.INC
444
<1> ; Last Modification: 12/07/2022 (Previous: 10/07/2015)
445
<1> ;
446
<1> ; DISK I/O SYSTEM INITIALIZATION - Erdogan Tan (Retro UNIX 386 v1 project)
447
<1> ;
448
<1> ; ////////// DISK I/O SYSTEM STRUCTURE INITIALIZATION //////////
449
<1> ;
450
<1> ; 09/08/2022
451
<1> ; 08/08/2022
452
<1> ; 14/07/2022 (TRDOS 386 v2.0.5)
453
<1> ; 12/07/2022 (Retro UNIX 386 v1.2)
454
<1> ; 29/08/2020
455
<1> ; 17/07/2020
456
<1> ; 14/07/2020 (TRDOS 386 v2.0.2)
457
<1> ; 10/12/2014 - 02/02/2015 - dsectrm2.s
458
459 ;L0:
460
<1> ; 12/11/2014 (Retro UNIX 386 v1 - beginning)
461
<1> ; Detecting disk drives... (by help of ROM-BIOS)
462
<1> mov     dx, 7Fh
463
L1:
464
<1> inc     dl
465
<1> mov     ah, 41h ; Check extensions present
466
<1> ; Phoenix EDD v1.1 - EDD v3
467
<1> mov     bx, 55AAh
468
<1> int     13h
469
<1> jc      short L2
470
471
<1> cmp     bx, 0AA55h
472
<1> jne     short L2
473
<1> inc     byte [hdc] ; count of hard disks (EDD present)
474
<1> mov     [last_drv], dl ; last hard disk number
475
<1> mov     bx, hd0_type - 80h
476
<1> add     bx, dx
477
<1> mov     [bx], cl ; Interface support bit map in CX
478
<1> ; Bit 0 - 1, Fixed disk access subset ready
479
<1> ; Bit 1 - 1, Drv locking and ejecting ready
480
<1> ; Bit 2 - 1, Enhanced Disk Drive Support
481
<1> ; (EDD) ready (DPTE ready)
482
<1> ; Bit 3 - 1, 64bit extensions are present
483
<1> ; (EDD-3)
484
<1> ; Bit 4 to 15 - 0, Reserved
485
486
<1> cmp     dl, 83h ; drive number < 83h
487
<1> jb      short L1
488
L2:
489
<1> ; 23/11/2014
490
<1> ; 19/11/2014
491
<1> xor     dl, dl ; 0
492
<1> ; 04/02/2016 (esi -> si)
493
<1> mov     si, fd0_type

```

```

68      <1> L3:
69      <1> ; 14/01/2015
70 00000127 8816[7966] <1> mov     [drv], dl
71      <1> ;
72 0000012B B408      <1> mov     ah, 08h ; Return drive parameters
73 0000012D CD13      <1> int     13h
74 0000012F 7210      <1> jc      short L4
75      <1> ; BL = drive type (for floppy drives)
76      <1> ; DL = number of floppy drives
77      <1> ;
78      <1> ; ES:DI = Address of DPT from BIOS
79      <1>
80 00000131 881C      <1> mov     [si], bl ; Drive type
81      <1> ; 4 = 1.44 MB, 80 track, 3 1/2"
82      <1> ; 14/01/2015
83 00000133 E8DD01    <1> call    set_disk_parms
84      <1> ; 10/12/2014
85 00000136 81FE[7C66] <1> cmp     si, fd0_type
86 0000013A 7705      <1> ja      short L4
87 0000013C 46        <1> inc     si ; fd1_type
88 0000013D B201      <1> mov     dl, 1
89 0000013F EBE6      <1> jmp     short L3
90      <1> L4:
91 00000141 B27F      <1> mov     dl, 7Fh
92      <1> ; 24/12/2014
93 00000143 803E[7B66]00 <1> cmp     byte [hdc], 0 ; EDD present or not ?
94      <1> ;ja      L10 ; yes, all fixed disk operations
95      <1> ; will be performed according to
96      <1> ; present EDD specification
97      <1> ; 14/07/2022
98 00000148 7603      <1> jna     short L5
99 0000014A E98B00    <1> jmp     L10
100     <1>
101     <1> L5:
102     <1> ; 17/07/2020
103     <1> ; Note: Virtual CPU will not come here while
104     <1> ; running in QEMU, Bochs, VirtualBox emulators !!!
105     <1>
106     <1> ; 17/07/2020
107     <1> ; Older BIOS (INT 13h, AH = 48h is not available)
108     <1>
109 0000014D FEC2      <1> inc     dl
110 0000014F 8816[7966] <1> mov     [drv], dl
111 00000153 8816[7A66] <1> mov     [last_drv], dl ; 14/01/2015
112 00000157 B408      <1> mov     ah, 08h ; Return drive parameters
113 00000159 CD13      <1> int     13h ; (conventional function)
114     <1> ;jc      L13 ; fixed disk drive not ready
115     <1> ; 14/07/2022
116 0000015B 7303      <1> jnc     short L6
117 0000015D E9A501    <1> jmp     L13
118     <1> L6:
119 00000160 8816[7B66] <1> mov     [hdc], dl ; number of drives
120     <1> ; 14/01/2013
121     <1> ;push cx
122 00000164 E8AC01    <1> call    set_disk_parms
123     <1> ;pop cx
124     <1> ;
125     <1> ;and cl, 3Fh ; sectors per track (bits 0-6)
126 00000167 8A16[7966] <1> mov     dl, [drv]
127 0000016B BB0401    <1> mov     bx, 65*4 ; hd0 parameters table (INT 41h)
128 0000016E 80FA80    <1> cmp     dl, 80h
129 00000171 7603      <1> jna     short L7
130 00000173 83C314    <1> add     bx, 5*4 ; hd1 parameters table (INT 46h)
131     <1> L7:
132 00000176 31C0      <1> xor     ax, ax
133 00000178 8ED8      <1> mov     ds, ax
134 0000017A 8B37      <1> mov     si, [bx]
135 0000017C 8B4702      <1> mov     ax, [bx+2]
136 0000017F 8ED8      <1> mov     ds, ax
137 00000181 3A4C0E    <1> cmp     cl, [si+FDPT_SPT] ; sectors per track
138     <1> ;jne     L12 ; invalid FDPT
139     <1> ; 14/07/2022
140 00000184 7403      <1> je      short L7_8
141 00000186 E97801    <1> jmp     L12
142     <1> L7_8:
143 00000189 BF0000    <1> mov     di, HD0_DPT
144 0000018C 80FA80    <1> cmp     dl, 80h
145 0000018F 7603      <1> jna     short L8
146 00000191 BF2000    <1> mov     di, HD1_DPT
147     <1> L8:
148     <1> ; 30/12/2014
149 00000194 B80090    <1> mov     ax, DPT_SEGM
150 00000197 8EC0      <1> mov     es, ax
151     <1> ; 24/12/2014
152 00000199 B90800    <1> mov     cx, 8
153 0000019C F3A5      <1> rep     movsw ; copy 16 bytes to the kernel's DPT location
154 0000019E 8CC8      <1> mov     ax, cs
155 000001A0 8ED8      <1> mov     ds, ax
156     <1>
157     <1> ; 02/02/2015
158     <1> ;mov     cl, [drv]
159     <1> ;mov     bl, cl
160     <1> ;mov     ax, 1F0h
161     <1> ;and     bl, 1
162     <1> ;jz      short L9
163     <1> ;shl     bl, 4
164     <1> ;sub     ax, 1F0h-170h
165     <1>
166     <1> ; 17/07/2020
167     <1> ; (Only 1F0h port address must be valid for old ROM BIOSes)
168 000001A2 B8F001    <1> mov     ax, 1F0h
169 000001A5 B3A0      <1> mov     bl, 0A0h
170 000001A7 80FA80    <1> cmp     dl, 80h
171 000001AA 7603      <1> jna     short L9
172     <1> ; dl = 81h
173 000001AC 80C310    <1> add     bl, 10h ; slave disk
174     <1> ;sub     ax, 1F0h-170h
175     <1> L9:
176 000001AF AB        <1> stosw   ; I/O PORT Base Address (1F0h, 170h)
177 000001B0 050602    <1> add     ax, 206h
178 000001B3 AB        <1> stosw   ; CONTROL PORT Address (3F6h, 376h)
179 000001B4 88D8      <1> mov     al, bl ; bit 4, master/slave disk bit
180     <1> ;add     al, 0A0h ; 17/07/2020
181 000001B6 AA        <1> stosb   ; Device/Head Register upper nibble
182     <1> ;
183 000001B7 FE06[7966] <1> inc     byte [drv]
184     <1> ;mov     bx, hd0_type - 80h
185     <1> ;add     bx, cx
186     <1> ; 09/08/2022 - BugFix
187 000001BB 30FF      <1> xor     bh, bh
188 000001BD 88D3      <1> mov     bl, dl
189 000001BF 81C3[FE65] <1> add     bx, hd0_type - 80h
190 000001C3 800F80    <1> or      byte [bx], 80h ; present sign (when lower nibble is 0)
191 000001C6 A0[7B66]  <1> mov     al, [hdc]

```

```

192 000001C9 FEC8      <1>      dec     al
193                   <1>      ;jz     L13
194                   <1>      ; 14/07/2022
195 000001CB 7408      <1>      jz      short L9_10
196 000001CD 80FA80    <1>      cmp     dl, 80h
197                   <1>      ;jna     L5 ; Max. 2 hard disks ; 17/07/2020
198                   <1>      ; 14/07/2022
199 000001D0 7703      <1>      ja      short L9_10
200 000001D2 E978FF    <1>      jmp     L5
201                   <1>      L9_10:
202 000001D5 E92D01    <1>      jmp     L13
203                   <1>      L10:
204 000001D8 FEC2      <1>      inc     dl
205                   <1>      ; 25/12/2014
206 000001DA 8816[7966] <1>      mov     [drv], dl
207 000001DE B408      <1>      mov     ah, 08h ; Return drive parameters
208 000001E0 CD13      <1>      int     13h ; (conventional function)
209                   <1>      ;jc     L13
210                   <1>      ; 14/07/2022
211 000001E2 72F1      <1>      jc      short L9_10
212                   <1>      ; 14/01/2015
213                   <1>      ;mov    dl, [drv]
214                   <1>      ; 09/08/2022
215                   <1>      ;push   dx
216 000001E4 51         <1>      push  cx
217 000001E5 E82B01    <1>      call  set_disk_parms
218 000001E8 59         <1>      pop    cx
219                   <1>      ;pop    dx
220                   <1>      ; 09/08/2022
221 000001E9 8A16[7966] <1>      mov     dl, [drv]
222                   <1>      ; 06/07/2016 (BugFix for >64K kernel files)
223                   <1>      ; 04/02/2016 (esi -> si)
224                   <1>      ;mov    si, _end ; 30 byte temporary buffer address
225                   <1>      ; ; at the '_end' of kernel.
226                   <1>      ;mov    word [si], 30
227                   <1>      ; 06/07/2016
228 000001ED BE[5400]   <1>      mov     si, _int13h_48h_buffer
229                   <1>      ; 09/07/2016
230 000001F0 B81E00    <1>      mov     ax, 001Eh
231 000001F3 8824      <1>      mov     [si], ah ; 0
232 000001F5 46        <1>      inc     si
233 000001F6 8904      <1>      mov     [si], ax
234                   <1>      ; word [si] = 30
235                   <1>      ;
236 000001F8 B448      <1>      mov     ah, 48h ; Get drive parameters (EDD function)
237 000001FA CD13      <1>      int     13h
238                   <1>      ;jc     L13
239                   <1>      ; 14/07/2022
240 000001FC 72D7      <1>      jc      short L9_10
241                   <1>      ;
242                   <1>      ; 29/08/2020
243                   <1>      ; 04/02/2016 (ebx -> bx)
244                   <1>      ; 14/01/2015
245 000001FE 28FF      <1>      sub     bh, bh
246 00000200 88D3      <1>      mov     bl, dl
247                   <1>      ;sub    bl, 80h
248                   <1>      ; 29/08/2020
249 00000202 81C3[FE65] <1>      add     bx, (hd0_type - 80h)
250                   <1>      ;mov    al, [bx]
251 00000206 8A07      <1>      mov     al, [bx]
252 00000208 0C80      <1>      or      al, 80h
253 0000020A 8807      <1>      mov     [bx], al
254 0000020C 81EB[7C66] <1>      sub     bx, hd0_type - 2 ; 15/01/2015
255                   <1>      ;add    bx, drv.status
256                   <1>      ;mov    [bx], al
257                   <1>      ; 29/08/2020
258 00000210 8887[9E66] <1>      mov     [bx+drv.status], al
259                   <1>      ; 04/02/2016 (eax -> ax)
260                   <1>      ;mov    ax, [si+16]
261                   <1>      ; 14/07/2020
262                   <1>      ;mov    di, [si+18]
263                   <1>      ;test   ax, [si+18]
264                   <1>      ;test   ax, di ; 14/07/2020
265                   <1>      ;jz     short L10_A0h ; (!) ; 17/07/2020
266                   <1>      ; ; 'CHS only' disks on EDD system
267                   <1>      ; ; are reported with ZERO disk size
268                   <1>      ; ; (if so, we must not overwrite
269                   <1>      ; ; calculated disk size in 'set_disk_parms')
270                   <1>      ; 29/08/2020
271 00000214 8B4410    <1>      mov     ax, [si+16]
272 00000217 8B7C12    <1>      mov     di, [si+18]
273 0000021A 09C0      <1>      or      ax, ax
274 0000021C 7504      <1>      jnz     short L10_LBA
275 0000021E 09FF      <1>      or      di, di
276 00000220 740B      <1>      jz      short L10_A0h
277                   <1>      L10_LBA:
278                   <1>      ;sub    bx, drv.status
279 00000222 C1E302    <1>      shl     bx, 2
280                   <1>      ;add    bx, drv.size ; disk size (in sectors)
281                   <1>      ;mov    [bx], ax
282                   <1>      ; 29/08/2020
283 00000225 8987[8266] <1>      mov     [bx+drv.size], ax
284                   <1>      ;mov    ax, [si+18]
285                   <1>      ;mov    [bx], ax
286                   <1>      ;mov    [bx+2], ax ; BugFix ; 15/07/2020
287                   <1>      ; 14/07/2020
288                   <1>      ;mov    [bx+2], di ; 15/07/2020
289                   <1>      ; 29/08/2020
290 00000229 89BF[8466] <1>      mov     [bx+drv.size+2], di
291                   <1>      L10_A0h:
292                   <1>      ; 17/07/2020
293                   <1>      ; Note: Virtual CPU will jump here from above (!) test
294                   <1>      ; while running in QEMU
295                   <1>      ;
296                   <1>      ; Jump here to fix a ZERO (LBA) disk size problem
297                   <1>      ; for CHS disks (28/02/2015)
298                   <1>      ;
299                   <1>      ; 30/12/2014
300 0000022D BF0000    <1>      mov     di, HD0_DPT
301 00000230 88D0      <1>      mov     al, dl
302 00000232 83E003    <1>      and     ax, 3
303 00000235 C0E005    <1>      shl     al, 5 ; * 32
304 00000238 01C7      <1>      add     di, ax
305 0000023A B80090    <1>      mov     ax, DPT_SEGM
306 0000023D 8EC0      <1>      mov     es, ax
307                   <1>      ;
308 0000023F 88E8      <1>      mov     al, ch ; max. cylinder number (bits 0-7)
309 00000241 88CC      <1>      mov     ah, cl
310 00000243 C0EC06    <1>      shr     ah, 6 ; max. cylinder number (bits 8-9)
311 00000246 40         <1>      inc     ax ; logical cylinders (limit 1024)
312 00000247 AB        <1>      stosw
313 00000248 88F0      <1>      mov     al, dh ; max. head number
314                   <1>      ;
315 0000024A 30F6      <1>      xor     dh, dh ; 29/08/2020 (dh = 0 is needed here)

```

```

316      <1>      ;
317 0000024C FEC0      <1>      inc     al
318 0000024E AA        <1>      stosb   ; logical heads (limits 256)
319 0000024F B0A0      <1>      mov     al, 0A0h ; Indicates translated table
320 00000251 AA        <1>      stosb
321 00000252 8A440C    <1>      mov     al, [si+12]
322 00000255 AA        <1>      stosb   ; physical sectors per track
323 00000256 31C0      <1>      xor     ax, ax
324      <1>      ;dec     ax ; 02/01/2015
325 00000258 AB        <1>      stosw   ; precompensation (obsolete)
326      <1>      ;xor     al, al ; 02/01/2015
327 00000259 AA        <1>      stosb   ; reserved
328 0000025A B008      <1>      mov     al, 8 ; drive control byte
329      <1>      ; (do not disable retries,
330      <1>      ; more than 8 heads)
331 0000025C AA        <1>      stosb
332 0000025D 8B4404    <1>      mov     ax, [si+4]
333 00000260 AB        <1>      stosw   ; physical number of cylinders
334      <1>      ;push    ax ; 02/01/2015
335 00000261 8A4408    <1>      mov     al, [si+8]
336 00000264 AA        <1>      stosb   ; physical num. of heads (limit 16)
337 00000265 29C0      <1>      sub     ax, ax
338      <1>      ;pop     ax ; 02/01/2015
339 00000267 AB        <1>      stosw   ; landing zone (obsolete)
340 00000268 88C8      <1>      mov     al, cl ; logical sectors per track (limit 63)
341 0000026A 243F      <1>      and     al, 3Fh
342 0000026C AA        <1>      stosb
343      <1>      ;sub     al, al ; checksum
344      <1>      ;stosb
345      <1>      ;
346 0000026D 83C61A    <1>      add     si, 26 ; (BIOS) DPTE address pointer
347 00000270 AD        <1>      lodsw
348 00000271 50        <1>      push    ax ; * ; (BIOS) DPTE offset
349 00000272 AD        <1>      lodsw
350 00000273 50        <1>      push    ax ; ** ; (BIOS) DPTE segment
351      <1>      ;
352      <1>      ; checksum calculation
353 00000274 89FE      <1>      mov     si, di
354 00000276 06        <1>      push    es
355 00000277 1F        <1>      pop     ds
356      <1>      ;mov     cx, 16
357 00000278 B90F00    <1>      mov     cx, 15
358 0000027B 29CE      <1>      sub     si, cx
359 0000027D 30E4      <1>      xor     ah, ah
360      <1>      ;del     cl
361      <1>      L11:
362 0000027F AC        <1>      lodsb
363 00000280 00C4      <1>      add     ah, al
364 00000282 E2FB      <1>      loop    L11
365      <1>      ;
366 00000284 88E0      <1>      mov     al, ah
367 00000286 F6D8      <1>      neg     al ; -x+x = 0
368 00000288 AA        <1>      stosb   ; put checksum in byte 15 of the tbl
369      <1>      ;
370 00000289 1F        <1>      pop     ds ; ** ; (BIOS) DPTE segment
371 0000028A 5E        <1>      pop     si ; * ; (BIOS) DPTE offset
372      <1>      ;
373      <1>      ; 08/08/2022 (TRDOS 386 v2.0.5)
374      <1>      ; (Recent version of Retro UNIX 386 v1 'diskinit.s' file
375      <1>      ; -12/07/2022- does not contain following 2020 code) (*)
376      <1>      ;
377      <1>      ; 14/07/2020 (TRDOS 386 v2.0.2)
378      <1>      ; 0FFFFh:0FFFFh = invalid DPTE address
379 0000028B 8B0C      <1>      mov     cx, [si]
380 0000028D 8B4402    <1>      mov     ax, [si+2]
381 00000290 21C1      <1>      and     cx, ax
382 00000292 41        <1>      inc     cx
383 00000293 7404      <1>      jz     short L11c ; 0FFFFh:0FFFFh
384 00000295 0B04      <1>      or     ax, [si]
385 00000297 752A      <1>      jnz    short L11e ; <> 0
386      <1>      L11c:
387      <1>      ; 17/07/2020
388      <1>      ; TRDOS 386 v2 DRVINIT assumptions:
389      <1>      ; (also by regarding QEMU, Bochs and VirtualBox settings)
390      <1>      ;
391      <1>      ; Hard disk 0 port address: 1F0h
392      <1>      ; Hard disk 1 port address: 1F0h
393      <1>      ; Hard disk 2 port address: 170h
394      <1>      ; Hard disk 3 port address: 170h
395      <1>      ; in QEMU, hda=hd0 (1F0h) and hdb=hd1 (1F0h) -IRQ14-
396      <1>      ; and hdc=hd2 (170h) and hdd=hd3 (170h) -IRQ15-
397      <1>      ;
398 00000299 B8F001    <1>      mov     ax, 1F0h
399      <1>      ;
400      <1>      ; 15/07/2020
401      <1>      ; 14/07/2020
402      <1>      ; Invalid DPTE address...
403      <1>      ; Default DPTE parms must be set for DISK_IO_CONT
404      <1>      ; (diskio.s)
405      <1>      ; 17/07/2020
406      <1>      ;
407      <1>      ;mov     bl, dl
408      <1>      ;and     bl, 1
409      <1>      ;jz     short L11d
410      <1>      ;
411 0000029C B3A0      <1>      mov     bl, 0A0h
412      <1>      ;
413 0000029E F6C201    <1>      test    dl, 1
414 000002A1 7403      <1>      jz     short L11g ; Master (as default, for 80h & 82h))
415      <1>      ;shl     bl, 4 ; bl = 16 (bit 4 = 1 -> slave)
416 000002A3 80C310    <1>      add     bl, 10h ; Slave (as default, for 81h & 83h)
417      <1>      L11g:
418      <1>      ; 17/07/2020
419 000002A6 80FA82    <1>      cmp     dl, 82h ; Hard disk 3 or 4 ?
420 000002A9 7203      <1>      jb     short L11d ; Primary ATA channel (hd0, hd1)
421      <1>      ; (port address = 1F0h)
422      <1>      ;
423      <1>      ; Secondary ATA channel (hd2, hd3)
424      <1>      ; (port address = 170h)
425      <1>      ;
426 000002AB 2D8000    <1>      sub     ax, 1F0h-170h
427      <1>      L11d:
428      <1>      ; 14/07/2020
429 000002AE AB        <1>      stosw   ; I/O PORT Base Address (1F0h, 170h)
430 000002AF 050602    <1>      add     ax, 206h
431 000002B2 AB        <1>      stosw   ; CONTROL PORT Address (3F6h, 376h)
432 000002B3 88D8      <1>      mov     al, bl ; Master/Slave bit (0 = Master)
433      <1>      ; 17/07/2020
434      <1>      ;or     al, 0A0h ; CHS (LBA enable bit = 0)
435      <1>      ; (Bits 5&7, reserved bits = 1)
436 000002B5 30E4      <1>      xor     ah, ah
437      <1>      ;stosb   ; Device/Head Register upper nibble
438 000002B7 AB        <1>      stosw
439 000002B8 30C0      <1>      xor     al, al

```

```

440 000002BA B90500 <1> mov cx, 5
441 000002BD F3AB <1> rep stosw ; clear remain part of the (fake) DPTE
442 000002BF 0E <1> push cs
443 000002C0 1F <1> pop ds
444 000002C1 EB2E <1> jmp short L11f
445 <1>
446 <1> ; 08/08/2022 (TRDOS 386 v2.0.5)
447 <1> ; (Recent version of Retro UNIX 386 v1 'diskinit.s' file
448 <1> ; -12/07/2022- does not contain above 2020 code) (*)
449 <1> L11e:
450 <1> ; 23/02/2015
451 000002C3 57 <1> push di
452 <1> ; ES:DI points to DPTE (FDPTE) location
453 <1> ; mov cx, 8
454 <1> ; mov cl, 8
455 000002C4 B90800 <1> mov cx, 8 ; 14/07/2020
456 000002C7 F3A5 <1> rep movsw
457 <1>
458 <1> ; 23/02/2015
459 <1> ; (P)ATA drive and LBA validation
460 <1> ; (invalidating SATA drives and setting
461 <1> ; CHS type I/O for old type fixed disks)
462 000002C9 5B <1> pop bx
463 000002CA 8CC8 <1> mov ax, cs
464 000002CC 8ED8 <1> mov ds, ax
465 000002CE 268B07 <1> mov ax, [es:bx]
466 000002D1 3DF001 <1> cmp ax, 1F0h
467 000002D4 7413 <1> je short L11a
468 000002D6 3D7001 <1> cmp ax, 170h
469 000002D9 740E <1> je short L11a
470 <1> ; invalidation
471 <1> ; (because base port address is not 1F0h or 170h)
472 <1> ; xor bh, bh
473 <1> ; mov bl, dl
474 <1> ; 29/08/2020
475 <1> ; xor dh, dh ; 0
476 000002DB 89D3 <1> mov bx, dx
477 <1> ; sub bl, 80h
478 <1> ; mov byte [bx+hd0_type], 0 ; not a valid disk drive !
479 <1> ; or byte [bx+drv.status+2], 0F0h ; (failure sign)
480 <1> ; 29/08/2020
481 000002DD C687[FE65]00 <1> mov byte [bx+hd0_type-80h], 0
482 000002E2 808F[2066]F0 <1> or byte [bx+drv.status-7Eh], 0F0h
483 000002E7 EB0F <1> jmp short L11b
484 <1> L11a:
485 <1> ; LBA validation
486 000002E9 268A4704 <1> mov al, [es:bx+4] ; Head register upper nibble
487 000002ED A840 <1> test al, 40h ; LBA bit (bit 6)
488 000002EF 7507 <1> jnz short L11b ; LBA type I/O is OK! (E0h or F0h)
489 <1> L11f:
490 <1> ; force CHS type I/O for this drive (A0h or B0h)
491 <1> ; sub bh, bh
492 <1> ; mov bl, dl
493 <1> ; 29/08/2020
494 <1> ; xor dh, dh ; 0
495 000002F1 89D3 <1> mov bx, dx
496 <1> ; sub bl, 80h ; 26/02/2015
497 <1> ; and byte [bx+drv.status+2], 0FEh ; clear bit 0
498 <1> ; ; bit 0 = LBA ready bit
499 <1> ; 29/08/2020
500 000002F3 80A7[2066]FE <1> and byte [bx+drv.status-7Eh], 0FEh
501 <1> ; 'diskio' procedure will check this bit !
502 <1> L11b:
503 000002F8 3A16[7A66] <1> cmp dl, [last_drv] ; 25/12/2014
504 000002FC 7307 <1> jnb short L13
505 000002FE E9D7FE <1> jmp L10
506 <1>
507 <1> L12:
508 <1> ; Restore data registers
509 00000301 8CC8 <1> mov ax, cs
510 00000303 8ED8 <1> mov ds, ax
511 <1> L13:
512 <1> ; 13/12/2014
513 00000305 0E <1> push cs
514 00000306 07 <1> pop es
515 <1> L14:
516 <1> ; clear keyboard buffer
517 00000307 B411 <1> mov ah, 11h
518 00000309 CD16 <1> int 16h
519 0000030B 7440 <1> jz short L16 ; no keys in keyboard buffer
520 0000030D B010 <1> mov al, 10h
521 0000030F CD16 <1> int 16h
522 00000311 EBF4 <1> jmp short L14
523 <1>
524 <1> set_disk_parms:
525 <1> ; 08/08/2022 - TRDOS 386 v2.0.5
526 <1> ; 09/05/2022 - Retro UNIX 386 v1.2
527 <1> ; 29/08/2020 - TRDOS 386 v2.0.2
528 <1> ; 04/02/2016 (ebx -> bx)
529 <1> ; 10/07/2015
530 <1> ; 14/01/2015
531 <1> ; push bx
532 00000313 28FF <1> sub bh, bh
533 00000315 8A1E[7966] <1> mov bl, [drv]
534 00000319 80FB80 <1> cmp bl, 80h
535 0000031C 7203 <1> jb short sdp0
536 0000031E 80EB7E <1> sub bl, 7Eh
537 <1> sdp0:
538 <1> ; add bx, drv.status
539 <1> ; mov byte [bx], 80h ; 'Present' flag
540 <1> ; 29/08/2020
541 00000321 C687[9E66]80 <1> mov byte [bx+drv.status], 80h
542 <1>
543 00000326 88E8 <1> mov al, ch ; last cylinder (bits 0-7)
544 00000328 88CC <1> mov ah, cl ;
545 0000032A C0EC06 <1> shr ah, 6 ; last cylinder (bits 8-9)
546 <1> ; sub bx, drv.status
547 0000032D D0E3 <1> shl bl, 1
548 <1> ; add bx, drv.cylinders
549 0000032F 40 <1> inc ax ; convert max. cyl number to cyl count
550 <1> ; mov [bx], ax
551 <1> ; 08/08/2022
552 <1> ; 29/08/2020
553 <1> ; mov [bx+drv.cylinders], ax
554 <1>
555 00000330 50 <1> push ax ; ** cylinders
556 <1> ; sub bx, drv.cylinders
557 <1> ; add bx, drv.heads
558 00000331 30E4 <1> xor ah, ah
559 00000333 88F0 <1> mov al, dh ; heads
560 00000335 40 <1> inc ax
561 <1> ; mov [bx], ax
562 <1> ; 08/08/2022
563 <1> ; 29/08/2020

```



```

564      <1>      ;mov     [bx+drv.heads], ax
565      <1>      ;sub     bx, drv.heads
566      <1>      ;add     bx, drv.spt
567 00000336 30ED      <1>      xor     ch, ch
568 00000338 80E13F    <1>      and     cl, 3Fh ; sectors (bits 0-6)
569      <1>      ;mov     [bx], cx
570      <1>      ; 08/08/2022
571      <1>      ; 29/08/2020
572      <1>      ;mov     [bx+drv.spt], cx
573      <1>      ;sub     bx, drv.spt
574 0000033B D1E3      <1>      shl     bx, 1
575      <1>      ;add     bx, drv.size ; disk size (in sectors)
576      <1>      ; LBA size = cylinders * heads * secpertrack
577 0000033D F7E1      <1>      mul     cx
578 0000033F 89C2      <1>      mov     dx, ax ; heads*spt
579 00000341 58        <1>      pop     ax ; ** cylinders
580      <1>      ; 09/05/2022 (fd0&fd1 drv.size = cyls*spt*heads)
581      <1>      ;dec     ax ; 1 cylinder reserved (!?) ; (*)
582 00000342 F7E2      <1>      mul     dx ; cylinders * (heads*spt)
583      <1>      ;mov     [bx], ax
584      <1>      ;mov     [bx+2], dx
585      <1>      ; 29/08/2020
586 00000344 8987[8266] <1>      mov     [bx+drv.size], ax
587 00000348 8997[8466] <1>      mov     [bx+drv.size+2], dx
588      <1>      ;
589      <1>      ;pop     bx
590 0000034C C3        <1>      retn
591      <1>
592      <1> L16: ; 28/05/2016
593
594      <1>      ; 10/11/2014
595      <1>      cli     ; Disable interrupts (clear interrupt flag)
596      <1>      ; Reset Interrupt MASK Registers (Master&Slave)
597      <1>      ;mov     al, 0FFh ; mask off all interrupts
598      <1>      ;out     21h, al ; on master PIC (8259)
599      <1>      ;jmp     $+2 ; (delay)
600      <1>      ;out     0A1h, al ; on slave PIC (8259)
601      <1>      ;
602      <1>      ; Disable NMI
603      <1>      mov     al, 80h
604      <1>      out     70h, al ; set bit 7 to 1 for disabling NMI
605      <1>      ;23/02/2015
606      <1>      ;nop
607      <1>      ;in      al, 71h ; read in 71h just after writing out to 70h
608      <1>      ; for preventing unknown state (!?)
609      <1>      ;
610      <1>      ; 20/08/2014
611      <1>      ; Moving the kernel 64 KB back (to physical address 0)
612      <1>      ; DS = CS = 1000h
613      <1>      ; 05/11/2014
614      <1>      xor     ax, ax
615      <1>      mov     es, ax ; ES = 0
616      <1>      ;
617      <1>      ; 04/07/2016 - TRDOS 386 (64K - 128K kernel)
618      <1>      xor     si, si
619      <1>      xor     di, di
620      <1>      ;mov     cx, 16384
621      <1>      ;rep     movsd
622      <1>      ; 02/12/2023
623      <1>      mov     cx, 32768
624      <1>      rep     movsw
625      <1>      ;
626      <1>      push    es ; 0
627      <1>      push    L17
628      <1>      retf
629
630      <1> L17:
631      <1>      mov     cx, 1000h
632      <1>      mov     es, cx ; 1000h
633      <1>      add     cx, cx
634      <1>      mov     ds, cx ; 2000h
635      <1>      ;sub     si, si
636      <1>      ;sub     di, di
637      <1>      ;mov     cx, 16384
638      <1>      ;rep     movsd
639      <1>      ; 02/12/2023
640      <1>      ; si = di = 0
641      <1>      mov     cx, 32768
642      <1>      rep     movsw
643
644      <1>      ; Turn off the floppy drive motor
645      <1>      mov     dx, 3F2h
646      <1>      out     dx, al ; 0 ; 31/12/2013
647
648      <1>      ; Enable access to memory above one megabyte
649
650      <1> L18:
651      <1>      in      al, 64h
652      <1>      test     al, 2
653      <1>      jnz     short L18
654      <1>      mov     al, 0D1h ; write output port
655      <1>      out     64h, al
656
657      <1> L19:
658      <1>      in      al, 64h
659      <1>      test     al, 2
660      <1>      jnz     short L19
661      <1>      mov     al, 0DFh ; Enable A20 line
662      <1>      out     60h, al
663
664      <1> ;L20:
665      <1>      ;
666      <1>      ; Load global descriptor table register
667
668      <1>      ;mov     ax, cs
669      <1>      ;mov     ds, ax
670
671      <1>      lgdt     [cs:gdt]
672
673      <1>      mov     eax, cr0
674      <1>      ;or      al, 1 ; 24/07/2023
675      <1>      inc     ax
676      <1>      mov     cr0, eax
677
678      <1>      ; Jump to 32 bit code
679
680      <1>      db 66h ; Prefix for 32-bit
681      <1>      db 0EAh ; Opcode for far jump
682      <1>      dd StartPM ; Offset to start, 32-bit
683      <1>      ; (1000h:StartPM = StartPM + 10000h)
684      <1>      ; This is the selector for CODE32_DESCRIPTOR,
685      <1>      ; assuming that StartPM resides in code32
686
687      <1>      ; 20/02/2017
688
689      <1> [BITS 32]

```

```

520      StartPM:
521      ; Kernel Base Address = 0 ; 30/12/2013
522      mov ax, KDATA      ; Save data segment identifier
523      mov ds, ax      ; Move a valid data segment into DS register
524      mov es, ax      ; Move data segment into ES register
525      mov fs, ax      ; Move data segment into FS register
526      mov gs, ax      ; Move data segment into GS register
527      mov ss, ax      ; Move data segment into SS register
528      ;mov esp, 90000h      ; Move the stack pointer to 090000h
529      ; 04/12/2023 - TRDOS 386 v2.0.7
530      ;mov esp, 97000h      ; 3072 bytes system stack (96400h-97000h)
531      mov esp, SYSTEMSTACK_ADDR ; 97000h (max. 3072 bytes)
532
533      clear_bss: ; Clear uninitialized data area
534      ; 11/03/2015
535      xor     eax, eax ; 0
536      mov     ecx, (bss_end - bss_start)/4
537      shr     ecx, 2 ; bss section is already aligned for double words
538      mov     edi, bss_start
539      rep     stosd
540
541      memory_init:
542      ; Initialize memory allocation table and page tables
543      ; 04/12/2023
544      ; 29/11/2023
545      ; 27/11/2023
546      ; 23/11/2023 (TRDOS 386 v2.0.7)
547      ; 24/07/2022 (TRDOS 386 v2.0.5)
548      ; 18/04/2021 (TRDOS 386 v2.0.4)
549      ; 16/11/2014
550      ; 15/11/2014
551      ; 07/11/2014
552      ; 06/11/2014
553      ; 05/11/2014
554      ; 04/11/2014
555      ; 31/10/2014 (Retro UNIX 386 v1 - Beginning)
556      ;
557      ; xor     eax, eax
558      ; xor     ecx, ecx
559      000003C0 B108      mov     cl, 8
560      000003C2 BF00001000 mov     edi, MEM_ALLOC_TBL
561      000003C7 F3AB      rep     stosd      ; clear Memory Allocation Table
562      ; for the first 1 MB memory
563      ;
564      000003C9 668B0D[74660000] mov     cx, [mem_1m_1k]      ; Number of contiguous KB between
565      ; 1 and 16 MB, max. 3C00h = 15 MB.
566      ;shr     cx, 2      ; convert 1 KB count to 4 KB count
567      ; 24/07/2022
568      000003D0 C1E902      shr     ecx, 2
569      000003D3 890D[A8760100] mov     [free_pages], ecx
570      000003D9 668B15[76660000] mov     dx, [mem_16m_64k] ; Number of contiguous 64 KB blocks
571      ; between 16 MB and 4 GB.
572      000003E0 6609D2      or     dx, dx
573      000003E3 7425      jz     short mi_0
574
575      ; 04/12/2023
576      %if 1
577      ; 02/12/2023 - temporary (2816MB limit)
578      ;cmp     dx, 44800 ; 0AF00h
579      000003E5 6681FA009F cmp     dx, 40704 ; (2560MB limit)
580      000003EA 7604      jna     short mi_x
581      ;mov     dx, 44800
582      000003EC 66BA009F      mov     dx, 40704
583      mi_x:
584      %endif
585      ; 23/11/2023 - temporary
586      ;and     dx, 3FFFh
587
588      000003F0 668915[76660000] mov     [mem_16m_64k], dx
589
590      000003F7 6689D0      mov     ax, dx
591      000003FA C1E004      shl     eax, 4      ; 64 KB -> 4 KB (page count)
592      000003FD 0105[A8760100] add     [free_pages], eax
593      00000403 0500100000 add     eax, 4096      ; 16 MB = 4096 pages
594      00000408 EB06      jmp     short mi_1
595      mi_0:
596      ;mov     ax, cx
597      ; 24/07/2022
598      0000040A 89C8      mov     eax, ecx
599      0000040C 66050001 add     ax, 256      ; add 256 pages for the first 1 MB
600      ;add     eax, 256
601      mi_1:
602      mov     [memory_size], eax ; Total available memory in pages
603      ; 1 alloc. tbl. bit = 1 memory page
604      ; 32 allocation bits = 32 mem. pages
605      ;
606      00000415 05FF7F0000 add     eax, 32767      ; 32768 memory pages per 1 M.A.T. page
607      0000041A C1E80F      shr     eax, 15      ; ((32768 * x) + y) pages (y < 32768)
608      ; --> x + 1 M.A.T. pages, if y > 0
609      ; --> x M.A.T. pages, if y = 0
610      0000041D 66A3[B8760100] mov     [mat_size], ax      ; Memory Alloc. Table Size in pages
611      00000423 C1E00C      shl     eax, 12      ; 1 M.A.T. page = 4096 bytes
612      ; Max. 32 M.A.T. pages (4 GB memory)
613      00000426 89C3      mov     ebx, eax      ; M.A.T. size in bytes
614      ; Set/Calculate Kernel's Page Directory Address
615      00000428 81C300001000 add     ebx, MEM_ALLOC_TBL
616      0000042E 891D[A0760100] mov     [k_page_dir], ebx ; Kernel's Page Directory address
617      ; just after the last M.A.T. page
618      ;
619      00000434 83E804      sub     eax, 4      ; convert M.A.T. size to offset value
620      00000437 A3[B0760100] mov     [last_page], eax ; last page offset in the M.A.T.
621      ; (allocation status search must be
622      ; stopped after here)
623      0000043C 31C0      xor     eax, eax
624      0000043E 48      dec     eax      ; FFFFFFFFh (set all bits to 1)
625      ;push     cx
626      ; 18/04/2021
627      0000043F 51      push    ecx
628      ; ecx = 3840 ; 27/11/2023
629      ; (Note: ecx < 3840 if the total memory is less than 16 MB)
630      00000440 C1E905      shr     ecx, 5      ; convert 1 - 16 MB page count to
631      ; count of 32 allocation bits
632      ; ecx = 120 ; 27/11/2023
633      00000443 F3AB      rep     stosd
634      ;pop     cx
635      ; 18/04/2021
636      00000445 59      pop     ecx
637      00000446 40      inc     eax      ; 0
638      00000447 80E11F      and     cl, 31      ; remain bits
639      0000044A 7412      jz     short mi_4
640      0000044C 8907      mov     [edi], eax      ; reset
641      mi_2:
642      0000044E 0FAB07      bts     [edi], eax      ; 06/11/2014
643      00000451 FEC9      dec     cl

```

```

644 00000453 7404      jz      short mi_3
645 00000455 FEC0      inc      al
646 00000457 EBF5      jmp      short mi_2
647
648 00000459 28C0      sub      al, al          ; 0
649 0000045B 83C704    add      edi, 4          ; 15/11/2014
650
651 0000045E 6609D2    or       dx, dx          ; check 16 MB to 4 GB memory space
652 00000461 7421      jz      short mi_6          ; max. 16 MB memory, no more...
653
654 00000463 B900021000    mov      ecx, MEM_ALLOC_TBL + 512 ; End of first 16 MB memory
655
656 00000468 29F9      sub      ecx, edi          ; displacement (to end of 16 MB)
657 0000046A 7406      jz      short mi_5          ; jump if EDI points to
658                                ; end of first 16 MB
659 0000046C D1E9      shr      ecx, 1          ; convert to dword count
660 0000046E D1E9      shr      ecx, 1          ; (shift 2 bits right)
661 00000470 F3AB      rep      stosd          ; reset all bits for reserved pages
662                                ; (memory hole under 16 MB)
663
664 00000472 6689D1    mov      cx, dx          ; count of 64 KB memory blocks
665 00000475 D1E9      shr      ecx, 1          ; 1 alloc. dword per 128 KB memory
666 00000477 9C          pushf          ; 16/11/2014
667 00000478 48          dec      eax          ; FFFFFFFFh (set all bits to 1)
668 00000479 F3AB      rep      stosd
669 0000047B 40          inc      eax          ; 0
670 0000047C 9D          popf          ; 16/11/2014
671 0000047D 7305      jnc      short mi_6
672 0000047F 6648      dec      ax          ; eax = 0000FFFFh
673 00000481 AB          stosd
674 00000482 6640      inc      ax          ; 0
675
676 00000484 39DF      cmp      edi, ebx          ; check if EDI points to
677 00000486 730A      jnb      short mi_7          ; end of memory allocation table
678                                ; (>= MEM_ALLOC_TBL + 4906)
679 00000488 89D9      mov      ecx, ebx          ; end of memory allocation table
680 0000048A 29F9      sub      ecx, edi          ; convert displacement/offset
681 0000048C D1E9      shr      ecx, 1          ; to dword count
682 0000048E D1E9      shr      ecx, 1          ; (shift 2 bits right)
683 00000490 F3AB      rep      stosd          ; reset all remain M.A.T. bits
684
685                                ; Reset M.A.T. bits in M.A.T. (allocate M.A.T. pages)
686 00000492 BA00001000    mov      edx, MEM_ALLOC_TBL
687                                ; sub      ebx, edx          ; Mem. Alloc. Tbl. size in bytes
688                                ; shr      ebx, 12          ; Mem. Alloc. Tbl. size in pages
689 00000497 668B0D[B8760100]    mov      cx, [mat_size] ; Mem. Alloc. Tbl. size in pages
690 0000049E 89D7      mov      edi, edx
691 000004A0 C1EF0F    shr      edi, 15          ; convert M.A.T. address to
692                                ; byte offset in M.A.T.
693                                ; (1 M.A.T. byte points to
694                                ; 32768 bytes)
695                                ; Note: MEM_ALLOC_TBL address
696                                ; must be aligned on 128 KB
697                                ; boundary!
698 000004A3 01D7      add      edi, edx          ; points to M.A.T.'s itself
699                                ; eax = 0
700 000004A5 290D[A8760100]    sub      [free_pages], ecx ; 07/11/2014
701
702 000004AB 0FB307    btr      [edi], eax          ; clear bit 0 to bit x (1 to 31)
703                                ; dec      bl
704 000004AE FEC9      dec      cl
705 000004B0 7404      jz      short mi_9
706 000004B2 FEC0      inc      al
707 000004B4 EBF5      jmp      short mi_8
708
709                                ;
710                                ; Reset Kernel's Page Dir. and Page Table bits in M.A.T.
711                                ; (allocate pages for system page tables)
712
713                                ; edx = MEM_ALLOC_TBL
714 000004B6 8B0D[A4760100]    mov      ecx, [memory_size] ; memory size in pages (PTes)
715 000004BC 81C1FF030000    add      ecx, 1023          ; round up (1024 PTes per table)
716 000004C2 C1E90A    shr      ecx, 10          ; convert memory page count to
717                                ; page table count (PDE count)
718
719 000004C5 51          push      ecx          ; (**) PDE count (<= 1024)
720
721 000004C6 41          inc      ecx          ; +1 for kernel page directory
722
723 000004C7 290D[A8760100]    sub      [free_pages], ecx ; 07/11/2014
724
725 000004CD 8B35[A0760100]    mov      esi, [k_page_dir] ; Kernel's Page Directory address
726 000004D3 C1EE0C    shr      esi, 12          ; convert to page number
727
728 000004D6 89F0      mov      eax, esi          ; allocation bit offset
729 000004D8 89C3      mov      ebx, eax
730 000004DA C1EB03    shr      ebx, 3          ; convert to alloc. byte offset
731 000004DD 80E3FC    and      bl, 0FCh          ; clear bit 0 and bit 1
732                                ; to align on dword boundary
733 000004E0 83E01F    and      eax, 31          ; set allocation bit position
734                                ; (bit 0 to bit 31)
735
736 000004E3 01D3      add      ebx, edx          ; offset in M.A.T. + M.A.T. address
737
738 000004E5 0FB303    btr      [ebx], eax          ; reset relevant bit (0 to 31)
739
740 000004E8 46          inc      esi          ; next page table
741 000004E9 E2EB      loop     mi_10          ; allocate next kernel page table
742                                ; (ecx = page table count + 1)
743
744 000004EB 59          pop      ecx          ; (**) PDE count (= pg. tbl. count)
745
746                                ; Initialize Kernel Page Directory and Kernel Page Tables
747
748                                ; Initialize Kernel's Page Directory
749 000004EC 8B3D[A0760100]    mov      edi, [k_page_dir]
750 000004F2 89F8      mov      eax, edi
751 000004F4 0C03      or       al, PDE_A_PRESENT + PDE_A_WRITE
752                                ; supervisor + read&write + present
753 000004F6 89CA      mov      edx, ecx          ; (**) PDE count (= pg. tbl. count)
754
755 000004F8 0500100000    mi_11: add      eax, 4096          ; Add page size (PGSZ)
756                                ; EAX points to next page table
757
758 000004FD AB          stosd
759 000004FE E2F8      loop     mi_11
760                                ; sub      eax, eax          ; Empty PDE
761                                ; mov      cx, 1024          ; Entry count (PGSZ/4)
762                                ; 29/11/2023
763 00000502 B504      mov      ch, 4          ; cx = 4*256 = 1024
764 00000504 29D1      sub      ecx, edx
765 00000506 7402      jz      short mi_12
766 00000508 F3AB      rep      stosd          ; clear remain (empty) PDEs
767
768                                ; Initialization of Kernel's Page Directory is OK, here.

```

```

768
769
770
771
772
773 0000050A 8B0D[A4760100]
774 00000510 89CA
775 00000512 B003
776
777
778 00000514 AB
779 00000515 0500100000
780 0000051A E2F8
781
782
783 0000051C 81E2FF030000
784 00000522 7408
785
786
787 00000524 B504
788
789
790 00000526 29D1
791 00000528 31C0
792 0000052A F3AB
793
794
795
796
797 0000052C 89F8
798
799
800 0000052E C1E80F
801 00000531 24FC
802
803 00000533 A3[B4760100]
804 00000538 A3[AC760100]
805
806
807
808
809
810
811
812
813 0000053D A1[A0760100]
814 00000542 0F22D8
815 00000545 0F20C0
816 00000548 0D00000080
817 0000054D 0F22C0
818
819
820 00000550 EA
821 00000551 [57050000]
822 00000555 0800
823
824
825
826
827
828
829
830 00000557 B9E8030000
831 0000055C BF00800B00
832
833
834 00000561 B800070007
835 00000566 F3AB
836
837
838
839
840
841
842
843 00000568 BE[4A390100]
844
845
846 0000056D BF00800B00
847
848
849
850
851
852
853
854
855
856
857 00000572 B40A
858
859
860 00000574 E8BE030000
861
862
863
864
865
866
867
868 00000579 B011
869 0000057B E620
870
871 0000057D E6A0
872
873 0000057F B020
874 00000581 E621
875
876 00000583 B028
877 00000585 E6A1
878
879 00000587 B004
880 00000589 E621
881
882 0000058B B002
883 0000058D E6A1
884
885 0000058F B001
886 00000591 E621
887
888 00000593 E6A1
889
890
891

```

```

mi_12:
; Initialize kernel's Page Tables
;
; (EDI points to address of page table 0)
; eax = 0
mov ecx, [memory_size] ; memory size in pages
mov edx, ecx ; (***)
mov al, PTE_A_PRESENT + PTE_A_WRITE
; supervisor + read&write + present

mi_13:
stosd
add eax, 4096
loop mi_13
; and dx, 1023 ; (***)
; 30/08/2023
and edx, 1023
jz short mi_14
; mov cx, 1024
; 30/08/2023
mov ch, 4 ; 4*256 = 1024
; sub cx, dx ; from dx (<= 1023) to 1024
; 24/07/2022
sub ecx, edx
xor eax, eax
rep stosd ; clear remain (empty) PTEs
; of the last page table

mi_14:
; Initialization of kernel's Page Tables is OK, here.
;
; mov eax, edi ; end of the last page table page
; ; (beginning of user space pages)

shr eax, 15 ; convert to M.A.T. byte offset
and al, 0FCh ; clear bit 0 and bit 1 for
; aligning on dword boundary
mov [first_page], eax
mov [next_page], eax ; The first free page pointer
; for user programs
; (Offset in Mem. Alloc. Tbl.)

;
; Linear/FLAT (1 to 1) memory paging for the kernel is OK, here.
;
;
; Enable paging
;
; mov eax, [k_page_dir]
mov cr3, eax
mov eax, cr0
or eax, 80000000h ; set paging bit (bit 31)
mov cr0, eax
; jmp KCODE:StartPMP

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

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

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

; 29/01/2016 (TRDOS 386 = TRDOS v2.0)
; mov esi, starting_msg
; ; 14/08/2015 (kernel version message will appear
; ; when protected mode and paging is enabled)
; mov edi, 0B8000h ; 27/08/2014

; 30/11/2020
; 14/11/2020 (TRDOS 386 v2.0.3)
; cmp byte [vbe3], 3 ; 03h
; jne short pkv_1
; ; mov ah, 0Bh ; Black background, light cyan forecolor
; ; Light red TRDOS 386 version text shows VBE3 is ready !
; mov ah, 0Ch ; Black background, light red forecolor
; jmp short pkv_2

; pkv_1:
; mov ah, 0Ah ; Black background, light green forecolor
; pkv_2:
; 20/08/2014
; call printk

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

; mov al, 0FFh ; mask off all interrupts for now
; out 21h, al

```

```

892             ;; jmp $+2
893             ;out    0A1h, al
894
895             ; 02/04/2015
896             ; 26/03/2015 System call (INT 30h) modification
897             ; DPL = 3 (Interrupt service routine can be called from user mode)
898             ;
899             ;; Linux (v0.12) source code by Linus Torvalds (1991)
900             ; setup_idt:
901             ;
902             ;;; 16/02/2015
903             ;;mov    dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
904             ; 21/08/2014 (timer_int)
905             mov     esi, ilist
906             lea     edi, [idt]
907             ; 26/03/2015
908             mov     ecx, 48          ; 48 hardware interrupts (INT 0 to INT 2Fh)
909             ; 02/04/2015
910             mov     ebx, 80000h
911             rp_sidt1:
912             lodsd
913             mov     edx, eax
914             mov     dx, 8E00h
915             mov     bx, ax
916             mov     eax, ebx          ; /* selector = 0x0008 = cs */
917             ; /* interrupt gate - dpl=0, present */
918             stosd   ; selector & offset bits 0-15
919             mov     eax, edx
920             stosd   ; attributes & offset bits 16-23
921             loop    rp_sidt1
922             ; 15/04/2016
923             ; TRDOS 386 (TRDOS v2.0) /// 32 software interrupts ///
924             ;mov     cl, 16          ; 16 software interrupts (INT 30h to INT 3Fh)
925             mov     cl, 32          ; 32 software interrupts (INT 30h to INT 4Fh)
926             rp_sidt2:
927             lodsd
928             and     eax, eax
929             jz      short rp_sidt3
930             mov     edx, eax
931             mov     dx, 0EE00h      ; P=1b/DPL=11b/01110b
932             mov     bx, ax
933             mov     eax, ebx          ; selector & offset bits 0-15
934             stosd
935             mov     eax, edx
936             stosd
937             loop    rp_sidt2
938             jmp     short sidt_OK
939             rp_sidt3:
940             mov     eax, ignore_int
941             mov     edx, eax
942             mov     dx, 0EE00h      ; P=1b/DPL=11b/01110b
943             mov     bx, ax
944             mov     eax, ebx          ; selector & offset bits 0-15
945             rp_sidt4:
946             stosd
947             xchg    eax, edx
948             stosd
949             xchg    edx, eax
950             loop    rp_sidt4
951             sidt_OK:
952             lidt    [idtd]
953             ;
954             ; TSS descriptor setup ; 24/03/2015
955             mov     eax, task_state_segment
956             mov     [gdt_tss0], ax
957             rol     eax, 16
958             mov     [gdt_tss1], al
959             mov     [gdt_tss2], ah
960             mov     word [tss.IOPB], tss_end - task_state_segment
961             00000614 00
962             ;
963             ; IO Map Base address (when this address points
964             ; to end of the TSS, CPU does not use IO port
965             ; permission bit map for RING 3 IO permissions,
966             ; access to any IO ports in ring 3 will be forbidden.)
967             ;
968             ;mov     [tss.esp0], esp ; TSS offset 4
969             mov     word [tss.ss0], KDATA ; TSS offset 8 (SS)
970             mov     ax, TSS          ; It is needed when an interrupt
971             ; occurs (or a system call -software INT- is requested)
972             ; while cpu running in ring 3 (in user mode).
973             ; (Kernel stack pointer and segment will be loaded
974             ; from offset 4 and 8 of the TSS, by the CPU.)
975             ltr     ax ; Load task register
976             ;
977             esp0_set0:
978             ; 29/11/2023 - Erdogan Tan
979             ; -----
980             ; If we read following -disabled- stack page setting code...
981             ; when the memory size >= 3GB, one of page tables with the stack page
982             ; at 4MB-4096 address. So, to leave stack pointer at 90000h is better/default.
983             ; (Problem may not appears for <= 2.5GB main memory but following code is also
984             ; defective because kernel stack page would be seen as unallocated in the M.A.T.
985             ; without by adding a memory allocation code.)
986             ;
987             ; 1st 4MB layout: 1MB kernel -base- reserved + max. 128 KB M.A.T. (at 100000h)
988             ; + Kernel's page directory (4KB) -just after the M.A.T.-
989             ; + Kernel's page tables (main memory size / 1024)
990             ; Note:
991             ; 1 or 2 additional kernel page table(s) may be needed for Linear Frame Buffer
992             ; .. but, it/they will not have to be contiguous with other kernel page tables.
993             ; -----
994             ; 27/11/2023 - TRDOS 386 v2.0.7
995             %if 0
996             ; 30/07/2015
997             mov     ecx, [memory_size] ; memory size in pages
998             shl     ecx, 12 ; convert page count to byte count
999             cmp     ecx, CORE ; beginning of user's memory space (400000h)
1000             ; (kernel mode virtual address)
1001             jna     short esp0_set1
1002             ;
1003             ; If available memory > CORE (end of the 1st 4 MB)
1004             ; set stack pointer to CORE
1005             ; (Because, PDE 0 is reserved for kernel space in user's page directory)
1006             ; (PDE 0 points to page table of the 1st 4 MB virtual address space)
1007             mov     ecx, CORE
1008             esp0_set1:
1009             mov     esp, ecx ; top of kernel stack (**tss.esp0**)
1010             %endif
1011             esp0_set_ok:
1012             ; 30/07/2015 (**tss.esp0**)
1013
1014

```

```

1015 0000061C 8925[3C760100]      mov     [tss.esp0], esp      ; 90000h ; 29/11/2023
1016                                ; <-- 97000h ; 04/12/2023 (max. 3072 bytes)
1017 00000622 66C705[40760100]10-   mov     word [tss.ss0], KDATA
1017 0000062A 00
1018                                ; 14/08/2015
1019                                ; 10/11/2014 (Retro UNIX 386 v1 - Erdogan Tan)
1020                                ;
1021                                ; cli      ; Disable interrupts (for CPU)
1022                                ; (CPU will not handle hardware interrupts, except NMI!)
1023                                ;
1024 0000062B 30C0                    xor     al, al      ; Enable all hardware interrupts!
1025 0000062D E621                    out     21h, al      ; (IBM PC-AT compatibility)
1026 0000062F EB00                    jmp     $+2        ; (All conventional PC-AT hardware
1027 00000631 E6A1                    out     0A1h, al     ; interrupts will be in use.)
1028                                ; (Even if related hardware component
1029                                ; does not exist!)
1030                                ; Enable NMI
1031 00000633 B07F                    mov     al, 7Fh     ; Clear bit 7 to enable NMI (again)
1032 00000635 E670                    out     70h, al
1033                                ; 23/02/2015
1034 00000637 90                      nop
1035 00000638 E471                    in      al, 71h     ; read in 71h just after writing out to 70h
1036                                ; for preventing unknown state (!?)
1037                                ;
1038                                ; Only a NMI can occur here... (Before a 'STI' instruction)
1039                                ;
1040                                ; 02/09/2014
1041                                ; xor     bx, bx
1042                                ; 24/07/2022
1043 0000063A 31DB                    xor     ebx, ebx
1044 0000063C 66BA0002                mov     dx, 0200h    ; Row 2, column 0 ; 07/03/2015
1045 00000640 E82E1D0000              call    _set_cpos    ; 24/01/2016
1046                                ;
1047                                ; 14/11/2020 (TRDOS 386 v2.0.3)
1048                                ; Check VBE3 protected mode interface/feature(s)
1049                                ;
1050                                ; cmp     byte [vbe3], 3 ; 03h
1051                                ; jne     short display_mem_info
1052                                ;
1053                                ; 20/11/2020
1054 00000645 803D[86090000]02        cmp     byte [vbe3], 2 ; 02h
1055 0000064C 7707                    ja      short vbe3_pmid_chk
1056                                ; jnb     short display_mem_info
1057                                ; jnb     display_mem_info ; 02/12/2020
1058 0000064E 7220                    jnb     short jmp_display_mem_info ; 24/07/2022
1059 00000650 E9EE010000              jmp     check_boch_plex86_vbe
1060                                ;
1061                                ; vbe3_pmid_chk:
1062 00000655 B9EA7F0000              mov     ecx, 32768 - (20+2) ; 32766 - PMInfoBlockSize
1063 0000065A BE02000C00              mov     esi, 0C0002h ; 1st word of the video bios rom is 0AA55h
1064                                ;
1065                                ; chk_pmi_sign:
1066                                ; mov     eax, [esi]
1067                                ; cmp     eax, 'PMID'
1068                                ; 30/11/2020
1069                                ; cmp     al, 'P'
1070                                ; jne     short chk_pmi_sign_next
1071 0000065F 813E504D4944              cmp     dword [esi], 'PMID'
1072                                ; je      short display_vbios_product_name
1073 00000665 740E                    je      short verify_pmib_chksum ; 15/11/2020
1074                                ; chk_pmi_sign_next:
1075                                ; inc     esi ; inc si
1076 00000668 E2F5                    loop    chk_pmi_sign
1077                                ;
1078                                ; not_valid_pmib:
1079 0000066A FE0D[86090000]          dec     byte [vbe3] ; 2 = VBE2 compatible
1080                                ; (vbe3 feature is defective in this vbios)
1081                                ; jmp     short display_mem_info
1082                                ; jmp_display_mem_info: ; 24/07/2022
1083                                ; 02/12/2020
1084 00000670 E987010000              jmp     display_mem_info
1085                                ;
1086                                ; verify_pmib_chksum:
1087                                ; 18/10/2023
1088                                ; (ATI RV370 video bios contains ZERO at Checksum offset.)
1089 00000675 31C0                    xor     eax, eax
1090 00000677 384613                  cmp     byte [esi+19], al ; 0 ; Checksum byte
1091 0000067A 7611                    jna     short skip_verify_pmib_chksum ; may be ATI video bios
1092                                ; cmp     dword [esi+16], 0C000h
1093                                ; ; CodeSegSel = 0C000h, InProtectMode = 0, Checksum = 0
1094                                ; je      short skip_verify_pmib_chksum ; may be ATI video bios
1095                                ; 18/10/2023
1096                                ; 15/11/2020
1097                                ; xor     eax, eax
1098                                ; mov     ecx, eax
1099                                ; mov     cl, 20
1100 0000067C 66B91400                mov     cx, 20 ; 30/11/2020
1101 00000680 56                      push    esi
1102                                ; pmib_sum_bytes:
1103                                ; lodsb
1104 00000681 AC                      add     ah, al
1105 00000682 00C4                    loop    pmib_sum_bytes
1106 00000684 E2FB                    pop     esi
1107 00000687 08E4                    or      ah, ah
1108 00000689 75DF                    jnz     short not_valid_pmib ; AH must be 0
1109                                ;
1110                                ; 18/10/2023
1111 0000068B 30C0                    xor     al, al ; eax = 0
1112                                ;
1113                                ; 28/02/2021
1114                                ; Set default (initial) truecolor bpp value to 32
1115                                ; (for VBE3 video bios.. because vbe3 video bioses
1116                                ; use 32bpp -for truecolor modes- instead of 24bpp)
1117                                ; (This setting may be changed via 'sysvideo' bx=0908h)
1118                                ;
1119                                ; skip_verify_pmib_chksum: ; 18/10/2023
1120                                ;
1121 0000068D C605[63730100]20        mov     byte [truecolor], 32 ; (RGB: 00RRGGBBh)
1122                                ;
1123                                ; display_vbios_product_name: ; 14/11/2020
1124                                ;
1125                                ; ; ESI points to 'PMID' (0C0000h + 'PMID' offset)
1126                                ;
1127                                ; 15/11/2020
1128                                ; mov     [pmid_addr], si ; PMInfoBlock offset
1129                                ; ; (in VGA bios, 0C0000h + offset)
1130                                ; 02/12/2020
1131                                ; push    esi ; * pmid_addr
1132 00000694 89F7                    mov     edi, esi
1133                                ;
1134                                ; mov     esi, [VBE3INFOBLOCK+22] ; 097E00h + 16h
1135                                ; ; OemVendorNamePtr (seg16:off16)
1136 00000696 8B35067E0900              mov     esi, [VBE3INFOBLOCK+6] ; 097E00h + 06h
1137                                ; ; OemStringPtr (seg16:off16)

```

```

1138 ; 18/10/2023
1139 ;xor    al, al ; eax = 0
1140 xchg   ax, si ; ax = offset, si = 0
1141 shr    esi, 12 ; (to convert segment to base addr)
1142 add    si, ax ; esi has an address < 1 MB limit
1143 ; (OemVendorName is in VBE3INFOBLOCK)
1144 ; Example:
1145 ; TRDOS 386 v2.0.3 VESA VBE3 protected mode
1146 ; interface development reference is ...
1147 ; NVIDIA GeForce FX5500 VGA BIOS -C000h:029Ch-
1148 ; Version 4.34.20.54.00 -C000h:02EDh-
1149 ; ((OemString is 'NVIDIA'))
1150 ; ((OemVendorName is 'NVIDIA Corporation'))
1151 ; ((OemProductName is 'NV34 Board - p162-1nz'))
1152
1153 ;mov    ah, 0Eh ; Black background, yellow forecolor
1154 ; 30/11/2020
1155 mov    ah, 0Ch ; Black background, light red forecolor
1156
1157 call   print_kmsg
1158
1159 ;mov    ah, 07h
1160
1161 mov    esi, vesa_vbe3_bios_msg
1162 ;call   print_kmsg
1163 call   pkmsg_loop ; 30/11/2020
1164
1165 ; 02/12/2020
1166 ;pop    edi ; * pmid_addr
1167
1168 ; 04/12/2023 - TRDOS 386 v2.0.7
1169 %if 0
1170 ; 24/07/2022
1171 ; 29/11/2020
1172 vbe3pminit:
1173 ; 30/11/2020
1174 ;cmp    byte [vbe3], 3 ; is VESA VBE3 PMI ready ?
1175 ;jne    short di4
1176
1177 ; Allocate 64KB contiguous (kernel) memory block
1178 xor     eax, eax
1179 mov     ecx, 65536
1180 call    allocate_memory_block
1181 ;jc     short di4
1182 ;jc     di0 ; 30/11/2020
1183 ; 24/07/2022
1184 jnc     short vbe3pminit0
1185 jmp     di0
1186
1187 vbe3pminit0:
1188 ; of course this block must be in the 1st 16MB
1189 ; because vbe3 pmi segments will be 16 bit segments
1190 ; (80286 type segment descriptors in GDT)
1191
1192 mov     [vbe3bios_addr], eax
1193 %endif
1194
1195 ; 04/12/2023 - TRDOS 386 v2.0.7 ; (!*!+)
1196 ; fixed PM-VBIOS address (no need to add a memory block)
1197 ; in the reserved -and already allocated- area under 1MB
1198 ; (purpose: to prevent page faults if memory size > 2.5GB)
1199 ; ((User's page dir contains only the 1st 4MB of system mem
1200 ; as PDE. So, this causes to page faults during an interrupt
1201 ; in user mode, because if memory size > 2.5GB, kernel
1202 ; page tables overs/passes 4MB limit and PM-VBIOS
1203 ; is located after kernel page tables.))
1204 vbe3pminit:
1205 mov     eax, VBE3BIOSCODE_ADDR ; 60000h ; (!*!+)
1206
1207 ; set [pmid_addr] to the new location
1208 mov     esi, 0C0000h
1209
1210 ; 30/11/2020
1211 sub     edi, esi ; izolate offset
1212 add     edi, eax ; new address
1213 mov     [pmid_addr], edi ; new 'PMID' location
1214
1215 ; Move VIDEO BIOS from 0C0000h to EAX
1216 mov     ecx, 65536/4
1217 mov     edi, eax ; 30/11/2020
1218 rep     movsd
1219
1220 ; 02/12/2020
1221 ; 30/11/2020
1222 ; set vbe3 segment selectors
1223
1224 ; 04/12/2023 - TRDOS 386 v2.0.7 ; (!*!+)
1225 %if 0
1226 ; VBE3CS (VESA VBE3 video bios code segment)
1227 mov     edi, _vbe3_CS+2 ; base address bits 0..15
1228 stosw   ; edi = _vbe3_CS+4
1229 ror     eax, 16
1230 mov     [edi], al ; base address, bits 16..23
1231
1232 ; VBE3DS ('CodeSegSel' in PMInfoBlock)
1233 mov     edi, _vbe3_DS+4 ; base addr bits 16..23
1234 mov     [edi], al
1235 rol     eax, 16
1236 mov     [edi-2], ax ; base address, bits 0..15
1237 %endif
1238 ; VBE3BDS (BIOSDataSel in PMInfoBlock)
1239 mov     edi, _vbe3_BDS+2 ; base addr bits 0..15
1240 mov     eax, VBE3BIOSDATABLOCK ; 1536 bytes
1241 stosw   ; edi = _vbe3_BDS+4
1242 shr     eax, 16
1243 mov     [edi], al ; base address, bits 16..23
1244
1245 ; VBE3SS (1024 bytes)
1246 mov     edi, _vbe3_SS+2 ; base addr bits 0..15
1247 mov     eax, VBE3STACKADDR ; size = 1024 bytes
1248 stosw   ; edi = _vbe3_SS+4
1249 shr     eax, 16
1250 mov     [edi], al ; base address, bits 16..23
1251
1252 ; stack pointer (esp) will be set to 1020
1253 ; (before VBE3 PMI call)
1254
1255 ; VBE3ES (max: 2048 bytes)
1256 mov     edi, _vbe3_ES+2 ; base addr bits 0..15
1257 mov     eax, VBE3SAVERESTOREBLOCK
1258 stosw   ; edi = _vbe3_ES+4
1259 shr     eax, 16
1260 mov     [edi], al ; base address, bits 16..23
1261

```

```

1262 ;Note: low word of _VBE3_ES base address will be
1263 ; set -again- by VBE3 PMI caller routine
1264
1265 ; 09/12/2020
1266 ;; set pmi32 (as VBE3 PMI is ready)
1267 ;inc byte [pmi32] ; = 1
1268
1269 ; KCODE16 (set PMI far return segment)
1270 mov edi, _16bit_CS+2 ; base addr bits 0..15
1271 mov eax, pminit_return_addr16
1272 stosw ; edi = _16bit_CS+4
1273 shr eax, 16
1274 mov [edi], al ; base address, bits 16..23
1275
1276 ; 30/11/2020
1277 ; clear mem from VBE3 BIOS data area emu block
1278 ; to end of vbe3 buffers
1279
1280 ; 18/10/2023 - TRDOS v2.0.7
1281 ; (VBE3BIOSDATABLOCK contains a copy of the 1st
1282 ; 1536 bytes of the memory, IVT, ROMBIOS DATA etc.)
1283 ; ((it must not be cleared here))
1284 %if 0
1285 ; 01/12/2020
1286 mov edi, VBE3BIOSDATABLOCK ; 97000h
1287 ;mov cx, (VBE3INFOBLOCK-VBE3BIOSDATABLOCK)/4
1288 ; ; ecx = 3584/4 double words
1289 ;
1290 ; 21/12/2020
1291 mov cx, (VBE3MODEINFOBLOCK-VBE3BIOSDATABLOCK)/4
1292 ; ecx = 3072/4 double words
1293 ;xor eax, eax
1294 xor al, al
1295 rep stosd
1296 %endif
1297 ; 18/10/2023 - TRDOS v2.0.7
1298 ; (VBE3BIOSDATABLOCK contains a copy of the 1st
1299 ; 1536 bytes of the memory, IVT, ROMBIOS DATA etc.)
1300 ; ((it must not be cleared here))
1301 mov edi, VBE3SAVERESTOREBLOCK ; 97600h
1302 mov cx, (VBE3MODEINFOBLOCK-VBE3SAVERESTOREBLOCK)/4
1303 ; ecx = 1536/4 = 384 double words
1304 ;xor eax, eax
1305 xor al, al
1306 rep stosd
1307
1308 ; Filling PMInfoBlock selector fields
1309 mov edi, [pmid_addr]
1310 mov word [edi+PMInfo.BIOSDataSel], VBE3BDS
1311 mov word [edi+PMInfo.A0000Sel], VBE3A000
1312 mov word [edi+PMInfo.B0000Sel], VBE3B000
1313 mov word [edi+PMInfo.B8000Sel], VBE3B800
1314 mov word [edi+PMInfo.CodeSegSel], VBE3DS
1315 mov byte [edi+PMInfo.InProtectMode], 1
1316
1317 ; Calculate and write checksum byte
1318 mov esi, edi
1319 mov cl, PMInfo.size - 1
1320 ;xor ah, ah
1321 pmid_chksum:
1322 lodsb
1323 add ah, al
1324 loop pmid_chksum
1325 neg ah ; 1 -> 255, 255 -> 1
1326 mov [esi], ah ; checksum
1327
1328 ; far call PM initialization
1329 ; (VBE3 video bios will return via 'retf')
1330 mov ax, [edi+PMInfo.PMInitialize]
1331 ; 30/11/2020
1332 shl eax, 16 ; save entry address in hw
1333 ; ax = 0
1334
1335 ; 02/12/2020
1336 push pminit_ok ; normal, near return address
1337
1338 ; 30/11/2020
1339 _VBE3PMI_fcall:
1340 ; ax = function, hw of eax = entry address
1341 pushf ; save 32 bit flags
1342 push esi ; *
1343 push ebp ; **
1344
1345 mov ebp, esp ; save 32 bit stack pointer
1346
1347 ;mov esi, eax
1348 ; 28/12/2025 (PMI32 BugFix)
1349 movzx esi, ax
1350
1351 cli
1352
1353 ; Disable interrupts (clear interrupt flag)
1354 ; Reset Interrupt MASK Registers (Master&Slave)
1355 mov al, 0FFh ; mask off all interrupts
1356 out 21h, al ; on master PIC (8259)
1357 jmp $+2 ; (delay)
1358 out 0A1h, al ; on slave PIC (8259)
1359
1360 ; 02/12/2020
1361 mov ax, VBE3SS
1362 mov ss, ax
1363
1364 mov esp, 1020 ; 30/11/2020
1365
1366 ; 01/12/2020
1367 ;lss esp, [stack16]
1368
1369 shr eax, 16 ; now, entry address is in 1w
1370
1371 ; 30/11/2020 - 16 bit pm selector test (OK)
1372 ; (32 bit stack push/pop & retf with 32 bit code segment)
1373 ; (16 bit stack push/pop with 16 bit code segment)
1374
1375 ; return
1376 ;push KCODE16
1377 ;push 0 ; 30/11/2020 (pminit_return_addr16)
1378
1379 ; 30/11/2020 (16 bit stack during retf from video bios)
1380 mov dword [esp], KCODE16 << 16
1381 ; ip = 0, cs = KCODE16
1382 ; 01/12/2020
1383 ;mov dword [VBE3STACKADDR+1020], KCODE16*65536
1384
1385 ;mov [jumpfar16], eax

```



```

1386
1387 ; 02/12/2020
1388 ; 30/11/2020 (32 bit stack during retf from kernel)
1389 ; far jump/call via retf
1390 0000078A 6A30 push VBE3CS ; VBE3 video bios's code segment
1391 0000078C 50 push eax ; PMInitialize or EntryPoint
1392
1393 ;mov ax, si ; restore function
1394 ; 24/07/2022
1395 0000078D 89F0 mov eax, esi
1396
1397 ; 02/12/2020
1398 0000078F 31F6 xor esi, esi ; (not necessary, it is not used)
1399
1400 00000791 CB retf ; far return (to 16 bit code segment)
1401
1402 ; 01/12/2020
1403 ;db 0EAh ; far jump to 16 bit code segment
1404 ;jumpfar16:
1405 ;dd 0
1406 ;dw VBE3CS
1407
1408 ;stack16:
1409 ;dd 1020
1410 ;dw VBE3SS
1411
1412 align 2
1413
1414 pminit_return_addr16:
1415 ; 02/12/2020
1416 ; 30/11/2020
1417 ;db 66h ; Prefix for 32-bit
1418 ;db 0EAh ; Opcode for far jump
1419 ;dd pminit_return_addr32 ; 32 bit Offset
1420 ;dw KCODE ; 32 bit code segment
1421 ; 01/12/2020
1422 00000792 EA[99070000]0800 jmp KCODE:pminit_return_addr32
1423
1424 pminit_return_addr32:
1425 ; restore 32 bit kernel selectors and 32 bit stack addr
1426 00000799 BE10000000 mov esi, KDATA
1427 0000079E 8EDE mov ds, si
1428 000007A0 8EC6 mov es, si
1429 000007A2 8ED6 mov ss, si
1430 000007A4 89EC mov esp, ebp ; top of stack = iretd return addr
1431
1432 000007A6 5D pop ebp ; **
1433 000007A7 5E pop esi ; *
1434 000007A8 9D popf ; restore 32 bit flags
1435
1436 ; enable interrupts
1437
1438 000007A9 FA cli
1439
1440 ; 21/12/2020
1441 000007AA 50 push eax
1442
1443 000007AB 30C0 xor al, al ; Enable all hardware interrupts!
1444 000007AD E621 out 21h, al ; (IBM PC-AT compatibility)
1445 000007AF EB00 jmp $+2 ; (All conventional PC-AT hardware
1446 000007B1 E6A1 out 0A1h, al ; interrupts will be in use.)
1447 ; (Even if related hardware component
1448 ; does not exist!)
1449 000007B3 58 pop eax
1450
1451 000007B4 FB sti
1452
1453 ; top of stack = return address
1454 ; ('pminit_ok' for PMinInit)
1455
1456 000007B5 C3 retn
1457
1458 pminit_ok:
1459 ; 03/12/2020
1460 ; (set [pmid_addr] to PMI entry point for next calls)
1461 000007B6 8305[149D0100]04 add dword [pmid_addr], PMInfo.EntryPoint ; + 4
1462
1463 ; 17/01/2021
1464 ; copy EDID data from temporary location to final address
1465 000007BD 803D[19420000]4F cmp byte [edid], 4Fh
1466 000007C4 7510 jne short vbe3h_chc1
1467 ;mov ecx, 32 ; 128 bytes, 32 dwords
1468 ; 24/07/2022
1469 000007C6 31C9 xor ecx, ecx
1470 000007C8 B120 mov cl, 32
1471 000007CA BE007D0900 mov esi, VBE3EDIDINFOBLOCK ; 97D00h
1472 000007CF BF[8C9C0100] mov edi, edid_info
1473 000007D4 F3A5 rep movsd
1474 ; 17/01/2021
1475 vbe3h_chc1:
1476 ; 16/01/2021
1477 ; 06/12/2020
1478 ; Save video mode 03h regs/dac/bios state
1479 ;
1480 ;mov ax, 4F04h ; VESA VBE Function 04h
1481 ; ; Save/Restore State
1482 ;sub dl, dl ; 0 = return buffer size
1483 ;mov cx, 0Fh ; ctrl/bios/dac/regs
1484 ;
1485 ;call int10h_32bit_pmi
1486 ; bx = number of 64-byte blocks to hold the state buff
1487 ;
1488 ;mov [vbe3stbufsize], bx
1489 ; 16/01/2021
1490 ;or word [vbe3stbsflags], 32768 ; set bit 15
1491 ;mov ax, bx
1492 ;shl ax, 6 ; * 64
1493 ;mov [vbestatebufsize+30], ax
1494 ;
1495 ; 06/12/2020
1496 ; check 'vbe3stbufsize' (it must be <= 32)
1497 ;
1498 ;cmp bx, 32
1499 ;ja short display_mem_info ; light red forecolor
1500 ;
1501 ; 16/01/2021
1502 ;or byte [vbe3stbsflags], 1 ; set bit 0
1503 ;
1504 ; 30/11/2020
1505 ; Change VESA VBE3 BIOS text color in order to give
1506 ; "VBE3 PMI initialization has been succeeded" meaning
1507
1508 000007D6 BE40810B00 mov esi, 0B8000h + 160*2 ; row 2
1509 000007DB 89F7 mov edi, esi

```

```

1510 vbe3h_chcl_next:
1511 lodsw
1512 cmp ah, 0Ch ; light red forecolor
1513 jne short display_mem_info
1514 mov ah, 0Eh ; yellow forecolor
1515 stosw
1516 jmp short vbe3h_chcl_next
1517
1518 di5:
1519 ; 18/10/2023 - TRDOS 386 v2.0.7
1520 ; ATI RV370 video BIOS (VESA VBE2)
1521 ; has PMI. (as described in VESA VBE3 specification)
1522 ;
1523 inc byte [vbe3] ; [vbe3] = 2 -> 3
1524 ; will be decreased to 2 again if 'PMID'
1525 ; signature will not be found.
1526 jmp vbe3_pmid_chk ; check for ATI video BIOS
1527
1528 di0:
1529 ; 30/11/2020
1530 ; Memory allocation error !
1531 mov byte [vbe3], 0 ; disable VBE3
1532
1533 display_mem_info:
1534 ; 19/12/2020
1535 ; temporary
1536 ; 24/11/2023
1537 cmp byte [vbe3], 2
1538 jb short dmi
1539 call default_lfb_info
1540
1541 dmi:
1542 ; 06/11/2014
1543 call memory_info
1544 ; 14/08/2015
1545 ; call getch ; 28/02/2015
1546
1547 ; 07/12/2023
1548 ; check EDID info for LCD monitor -screen resolution-
1549 ; for modifying VGA mode 13h CRTC parameters
1550 ; (if it is needed or not)
1551 call video_mode_13h_parms
1552
1553 drv_init:
1554 sti ; Enable Interrupts
1555 ; 06/02/2015
1556 mov edx, [hd0_type] ; hd0, hd1, hd2, hd3
1557 mov bx, [fd0_type] ; fd0, fd1
1558 ; 22/02/2015
1559 and bx, bx
1560 jnz short di1
1561 ;
1562 or edx, edx
1563 jnz short di2
1564 ;
1565 setup_error:
1566 mov esi, setup_error_msg
1567
1568 psem:
1569 lodsb
1570 or al, al
1571 ;jz short haltx ; 22/02/2015
1572 jz short di3
1573 push esi
1574 ; 13/05/2016
1575 mov ebx, 7 ; Black background,
1576 ; light gray forecolor
1577 ; video page 0 (BH=0)
1578 call _write_tty
1579 pop esi
1580 jmp short psem
1581
1582 check_bochs_plex86_vbe:
1583 ; 20/10/2023
1584 ; 18/10/2023
1585 ; 20/11/2020
1586 ; check Bochs/Plex86 VGABios VBE extension
1587 ; (check if TRDOS 386 v2 is running on emulators)
1588 ; BOCHS/QEMU/VIRTUALBOX
1589 ;
1590 ; ref: vbe_display_api.txt
1591
1592 ; bochs/plex86 VGABios VBE source code
1593 ; by Jeroen Janssen (2002)
1594 ; and Volker Ruppert (2003-2020)
1595
1596 ; 20/10/2023
1597 ; (ATI RV370 video bios has PMI support but
1598 ; it is a VESA VBE2 bios. So, even if [vbe3] is set
1599 ; to 3 for PMI functionality, it is better to display
1600 ; VESA VBE number as it is declared by the video bios.)
1601 mov byte [vbe_vnumber], "2" ; 20/10/2023
1602
1603 sub eax, eax ; 0
1604 mov dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
1605 out dx, ax ; VBE_DISPI_INDEX_ID register
1606 ;mov ax, 0B0C0h ; VBE_DISPI_ID0
1607 ;mov dx, 1CFh ; VBE_DISPI_IOPORT_DATA
1608 inc dx
1609 ;out dx, ax
1610 ;nop
1611 in ax, dx
1612 cmp ah, 0B0h
1613 ;jne short not_boch_qemu_vbe
1614 ; 18/10/2023
1615 ;jne short display_mem_info
1616 jne short di5 ; ATI VESA VBE2 bios or another
1617
1618 cmp al, 0C5h ; it must be 0B0C4h or 0B0C5h ..
1619 ;ja short not_boch_qemu_vbe
1620 ja short display_mem_info
1621 cmp al, 0C0h ; 0B0C0h to 0B0C5h .. ; Qemu
1622 ;cmp al, 0C4h ; 0B0C4h or 0B0C5h is OK ; Bochs
1623 ;jb short not_boch_qemu_vbe
1624 jb short display_mem_info
1625
1626 ; save VESA VBE2 bios (bochs/qemu) signature
1627 ; for enabling VBE2 functions in TRDOS 386 v2 kernel
1628 mov [vbe2bios], al ; 0C4h or 0C5h (for BOCHS)
1629 ; (0C0h-0C5h for QEMU)
1630 ; 20/10/2023
1631 ;mov byte [vbe_vnumber], "2"
1632
1633 ; 26/11/2020

```

```

1634 ; "BOCHS/QEMU/VIRTUALBOX VBE2 Video BIOS ..".
1635 00000868 BE[30730100] mov esi, vbe2_bochs_vbios ; BOCH/QEMU vbios msg
1636 0000086D B40E mov ah, 0Eh ; Yellow font
1637 0000086F E80D390000 call print_kmsg
1638
1639 ; this is not necessary ! (20/11/2020)
1640 00000874 803D[87090000]C4 cmp byte [vbe2bios], 0C4h
1641 ;jb display_mem_info ; (QEMU)
1642 ; 02/12/2023
1643 0000087B 720E jnb short not_boch_qemu_vbe
1644
1645 ; Display kernel version message if 0E9h hack port
1646 ; is enabled (bochs emulator feature)
1647 0000087D 66BAE900 mov dx, 0E9h ; hack port for BOCHS
1648 00000881 BE[89730100] mov esi, kernel_version_msg
1649 kvmsg_next_char:
1650 00000886 AC lodsb
1651 00000887 08C0 or al, al
1652 00000889 7505 jnz short put_kvmsg_in_hack_port
1653 not_boch_qemu_vbe:
1654 0000088B E96CFFFFFF jmp display_mem_info
1655 put_kvmsg_in_hack_port:
1656 00000890 EE out dx, al
1657 00000891 EBF3 jmp short kvmsg_next_char
1658
1659 di1:
1660 ; supress 'jmp short T6'
1661 ; (activate fdc motor control code)
1662 00000893 66C705[E8090000]90- mov word [T5], 9090h ; nop
1662 0000089B 90
1663
1664 ;mov ax, int_0Eh ; IRQ 6 handler
1665 ;mov di, 0Eh*4 ; IRQ 6 vector
1666 ;stosw
1667 ;mov ax, cs
1668 ;stosw
1669 ; 16/02/2015
1670 ;mov dword [DISKETTE_INT], fdc_int ; IRQ 6 handler
1671
1672 0000089C E815460000 CALL DS_KETTE_SETUP ; Initialize Floppy Disks
1673
1674 000008A1 09D2 or edx, edx
1675 000008A3 740C jz short di3
1676
1677 000008A5 E842460000 di2: call DISK_SETUP ; Initialize Fixed Disks
1678 ;jc setup_error
1679 ; 24/07/2022
1680 000008AA 7305 jnc short di3
1681 000008AC E97AFFFFFF jmp setup_error
1682
1683 000008B1 E846380000 di3: call setup_rtc_int ; 22/05/2015 (dsectrpm.s)
1684
1685 000008B6 E8C92E0100 call display_disks ; 07/03/2015 (Temporary)
1686
1687 ;haltx:
1688 ; 14/08/2015
1689 ;call getch ; 22/02/2015
1690 ;sti ; Enable interrupts (for CPU)
1691 ; 29/01/2016
1692 ; sub ah, ah ; read time count
1693 ; call int1Ah
1694 ; mov edx, ecx ; 18.2 * seconds
1695 ;md_info_msg_wait1:
1696 ; 29/01/2016
1697 ; mov ah, 1
1698 ; call int16h
1699 ; jz short md_info_msg_wait2
1700 ; xor ah, ah ; 0
1701 ; call int16h
1702 ; jmp short md_info_msg_ok
1703 ;md_info_msg_wait2:
1704 ; sub ah, ah ; read time count
1705 ; call int1Ah
1706 ; cmp edx, ecx ; 18.2 * seconds
1707 ; jna short md_info_msg_wait3
1708 ; xchg edx, ecx
1709 ;md_info_msg_wait3:
1710 ; sub ecx, edx
1711 ; cmp ecx, 127 ; 7 seconds (18.2 * 7)
1712 ; jnb short md_info_msg_wait1
1713 ;md_info_msg_ok:
1714
1715 ; 15/12/2020
1716 ; set initial values of LFB parameters
1717 000008BB 803D[86090000]02 cmp byte [vbe3], 2
1718 000008C2 7225 jnb short di4
1719
1720 ;mov ax, [def_LFB_addr]
1721 ;shl eax, 16
1722 ;mov [LFB_ADDR], eax
1723 ;mov eax, 1024*768*3
1724 ;mov [LFB_SIZE], eax
1725
1726 000008C4 BEFE7B0900 mov esi, VBE3MODEINFOBLOCK - 2
1727 000008C9 66C7061801 mov word [esi], 0118h ; default vbe mode
1728 ; 1024*768, 24bpp
1729 000008CE E8D4300000 call set_lfbinfo_table
1730
1731 ;;;
1732 ; 28/11/2023
1733 ; 20/10/2023 - TRDOS 386 v2.0.7
1734 000008D3 8B35[1C9D0100] mov esi, [LFB_ADDR] ; LFB base address (in bytes)
1735 ;mov edx, [LFB_SIZE] ; 28/11/2023
1736 ;add edx, 4095
1737 ;;;
1738 ;mov edx, ((1024*768*4)+4095)>>12 ; 28/11/2023
1739 000008D9 BAE9070000 mov edx, ((1920*1080*4)+4095)>>12 ; 28/11/2023
1740 ; edx = LFB size in pages
1741
1742 ; 29/11/2023
1743 ;mov ebx, esi
1744 ;add ebx, 4095 ; LFB start addr is always in page boundary
1745 ;shr ebx, 12 ; convert byte address to page address
1746 000008DE 8B0D[A4760100] mov ecx, [memory_size]
1747 ;cmp ebx, ecx
1748 ;jnb short di4 ; LFB addr >= main memory size
1749
1750 ; (set the overlapped pages as allocated for kernel)
1751 ;;;
1752 ; 28/11/2023
1753 ; (and set all of LFB pages in the kernel's page tables)
1754
1755 000008E4 E896590000 call allocate_lfb_pages_for_kernel
1756
1757 di4:

```

```

1757 ; 08/09/2016
1758 000008E9 0F20C0 mov     eax, cr0
1759 000008EC A810 test    al, 10h ; Bit 4, ET (Extension Type)
1760 000008EE 7408 jz      short sysinit
1761 ; 27/02/2017
1762 000008F0 FE05[60830100] inc     byte [fpready]
1763 ; 80387 (FPU) is ready
1764 000008F6 DBE3 fninit ; Initialize Floating-Point Unit
1765 sysinit:
1766 ; 30/06/2015
1767 000008F8 E88D630000 call    sys_init
1768 ;
1769 ; jmp     cpu_reset ; 22/02/2015
1770 hang:
1771 ; 23/02/2015
1772 ; sti ; Enable interrupts
1773 000008FD F4 hlt
1774 ;
1775 ; nop
1776 ; ; 03/12/2014
1777 ; ; 28/08/2014
1778 ; mov     ah, 11h
1779 ; call    getc
1780 ; jz      _c8
1781 ;
1782 ; 23/02/2015
1783 ; 06/02/2015
1784 ; 07/09/2014
1785 000008FE 31DB xor     ebx, ebx
1786 00000900 8A1D[CE760100] mov     bl, [ptty] ; active_page
1787 00000906 89DE mov     esi, ebx
1788 ; shl     si, 1
1789 ; 24/07/2022
1790 00000908 D1E6 shl     esi, 1
1791 0000090A 81C6[D0760100] add     esi, ttychr
1792 00000910 668B06 mov     ax, [esi]
1793 00000913 6621C0 and     ax, ax
1794 ; jz      short _c8
1795 00000916 74E5 jz      short hang
1796 00000918 66C7060000 mov     word [esi], 0
1797 0000091D 80FB03 cmp     bl, 3 ; video page 3
1798 ; jb      short _c8
1799 00000920 72DB jb      short hang
1800 ;
1801 ; 13/05/2016
1802 ; 07/09/2014
1803 nxtl:
1804 ; push     bx
1805 ; 18/04/2021
1806 00000922 53 push    ebx
1807 00000923 66BB0E00 mov     bx, 0Eh ; Yellow character
1808 ; ; on black background
1809 ; bh = 0 (video page 0)
1810 ; Retro UNIX 386 v1 - Video Mode 0
1811 ; (PC/AT Video Mode 3 - 80x25 Alpha.)
1812 ; push     ax
1813 ; 18/04/2021
1814 00000927 50 push    eax
1815 00000928 E8AE190000 call    _write_tty
1816 ; pop     ax
1817 ; 18/04/2021
1818 0000092D 58 pop     eax
1819 ; pop     bx
1820 0000092E 5B pop     ebx
1821 0000092F 3C0D cmp     al, 0Dh ; carriage return (enter)
1822 ; jne     short _c8
1823 00000931 75CA jne     short hang
1824 00000933 B00A mov     al, 0Ah ; next line
1825 00000935 EBEB jmp     short nxtl
1826 ;
1827 ; _c8:
1828 ; ; 25/08/2014
1829 ; cli ; Disable interrupts
1830 ; mov     al, [scounter + 1]
1831 ; and     al, al
1832 ; jnz     hang
1833 ; call    rtc_p
1834 ; jmp     hang
1835 ;
1836 ; 27/08/2014
1837 ; 20/08/2014
1838 printk:
1839 ; mov     edi, [scr_row]
1840 pk1:
1841 00000937 AC lodsb
1842 00000938 08C0 or      al, al
1843 0000093A 7404 jz      short pkr
1844 0000093C 66AB stosw
1845 0000093E EBF7 jmp     short pk1
1846 pkr:
1847 00000940 C3 retn
1848 ;
1849 ; *****
1850 video_mode_13h_parms:
1851 ; 07/12/2023 - TRDOS 386 v2.0.7
1852 ; Check EDID information for LCD monitors
1853 ; and change mode 13h parameters if it is required
1854 ; (if resolution > 1280x1024, it is LCD/panel monitor
1855 ; and Video Mode 13h parameters will be modified for 60HZ
1856 ; because LCD monitors does/can not display 320x200 70HZ
1857 ; standard VGA mode)
1858 ;
1859 00000941 803D[19420000]4F cmp     byte [edid], 4Fh
1860 00000948 753B jne     short CRT_monitor
1861 ;
1862 0000094A BE[8C9C0100] mov     esi, edid_info
1863 0000094F 83C626 add     esi, 26h ; EDID Standard Timing Identification
1864 00000952 B908000000 mov     ecx, 8
1865 ;
1866 00000957 66AD lodsw
1867 00000959 3C81 cmp     al, 129 ; (1280/8)-31
1868 0000095B 770E ja      short LCD_monitor ; 16:9
1869 0000095D 3C01 cmp     al, 1
1870 0000095F 7624 jna     short CRT_monitor
1871 00000961 E2F4 loop   chk_edid
1872 ; jmp     short CRT_monitor
1873 00000963 C3 retn
1874 ;
1875 00000964 0B3EB9858FB8E2 mode13h_crtc_60hz:
1876 db      0Bh, 3Eh, 0B9h, 85h, 8Fh, 0B8h, 0E2h
1877 LCD_monitor:
1878 ; modify default (CRTC register) parameters for 60HZ
1879 ; (320x200 letterbox type screen will appear on LCD screen)
1880 ;
1881 ; Note: when/while [PMI32] -protected mode video bios-

```

```

1881 ; feature is enabled -by user- for MODE 13h, internal
1882 ; (TRDOS 386) VGA -video mode setting- parameters
1883 ; will not be used (will be bypassed)
1884 ; by TRDOS 386 kernel for all standard VGA modes.
1885 ; ((NVIDIA/ATI Video Bios PMI will be used.))
1886 ;
1887 ; ref: github, juj/60hz.cpp
1888 ; https://gist.github.com/juj/
1889 ;
1890 0000096B B107 mov cl, 7
1891 0000096D BE[64090000] mov esi, mode13h_crtc_60hz
1892 00000972 BF[BA690000] mov edi, vga_mode_13h+10+6 ; CRTC Registers (index: 6)
1893 00000977 A4 movsb ; Vertical Total Register
1894 00000978 A4 movsb ; Overflow Register
1895 00000979 BF[C4690000] mov edi, vga_mode_13h+10+16 ; CRTC Regs (index: 16)
1896 0000097E A4 movsb ; Vertical Retrace Start Register
1897 0000097F A4 movsb ; Vertical Retrace End Register
1898 00000980 A4 movsb ; Vertical Display Enable/End Register
1899 00000981 47 inc edi
1900 00000982 47 inc edi
1901 00000983 A4 movsb ; Vertical Blanking Start Register
1902 00000984 A4 movsb ; Vertical Blanking End Register
1903 CRT_monitor:
1904 00000985 C3 ret
1905 ; *****
1906 ;
1907 ; 14/11/2020 (TRDOS 386 v2.0.3)
1908 00000986 00 vbe3: db 0 ; VESA VBE version (must be 03h)
1909 ; for using video bios calls in protected mode
1910 vbe2bios:
1911 00000987 B0 db 0B0h ;
1912 ;pmid_addr:
1913 ;dw 0 ; > 0 if 'PMID' sign is found
1914 ; ; 'C'pmid' offset addr in VGA bios seg, 0C000h)
1915 ; ; 02/12/2020
1916 ;dw 0 ; 32 bit address in pmid_addr
1917 ;
1918 ; 28/02/2017
1919 ; 22/01/2017
1920 ; 15/01/2017
1921 ; 14/01/2017
1922 ; 02/01/2017
1923 ; 25/12/2016
1924 ; 19/12/2016
1925 ; 10/12/2016 (callback)
1926 ; 06/06/2016
1927 ; 23/05/2016
1928 ; 22/05/2016 - TRDOS 386 (TRDOS v2.0) Timer Event Modifications
1929 ; 25/07/2015
1930 ; 14/05/2015 (multi tasking -time sharing- 'clock', x_timer)
1931 ; 17/02/2015
1932 ; 06/02/2015 (unix386.s)
1933 ; 11/12/2014 - 22/12/2014 (dsectrm2.s)
1934 ;
1935 ; IBM PC-XT Model 286 Source Code - BIOS2.ASM (06/10/85)
1936 ;
1937 ;--- HARDWARE INT 08 H - ( IRQ LEVEL 0 ) -----
1938 ; THIS ROUTINE HANDLES THE TIMER INTERRUPT FROM FROM CHANNEL 0 OF :
1939 ; THE 8254 TIMER. INPUT FREQUENCY IS 1.19318 MHZ AND THE DIVISOR :
1940 ; IS 65536, RESULTING IN APPROXIMATELY 18.2 INTERRUPTS EVERY SECOND. :
1941 ; :
1942 ; THE INTERRUPT HANDLER MAINTAINS A COUNT (40:6C) OF INTERRUPTS SINCE :
1943 ; POWER ON TIME, WHICH MAY BE USED TO ESTABLISH TIME OF DAY. :
1944 ; THE INTERRUPT HANDLER ALSO DECREMENTS THE MOTOR CONTROL COUNT (40:40) :
1945 ; OF THE DISKETTE, AND WHEN IT EXPIRES, WILL TURN OFF THE :
1946 ; DISKETTE MOTOR(S), AND RESET THE MOTOR RUNNING FLAGS. :
1947 ; THE INTERRUPT HANDLER WILL ALSO INVOKE A USER ROUTINE THROUGH :
1948 ; INTERRUPT 1CH AT EVERY TIME TICK. THE USER MUST CODE A :
1949 ; ROUTINE AND PLACE THE CORRECT ADDRESS IN THE VECTOR TABLE. :
1950 ;-----
1951 ;
1952 ;
1953 timer_int: ; IRQ 0
1954 ;int_08h: ; Timer
1955 ; 14/10/2015
1956 ; Here, we are simulating system call entry (for task switch)
1957 ; (If multitasking is enabled,
1958 ; 'clock' procedure may jump to 'sysrelease')
1959 ;
1960 00000988 1E push ds
1961 00000989 06 push es
1962 0000098A 0FA0 push fs
1963 0000098C 0FA8 push gs
1964 ;
1965 0000098E 60 pushad ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
1966 0000098F 66B91000 mov cx, KDATA
1967 00000993 8ED9 mov ds, cx
1968 00000995 8EC1 mov es, cx
1969 00000997 8EE1 mov fs, cx
1970 00000999 8EE9 mov gs, cx
1971 ;
1972 0000099B 0F20D9 mov ecx, cr3
1973 0000099E 890D[648F0100] mov [cr3reg], ecx ; save current cr3 register value/content
1974 ;
1975 ; 14/01/2017
1976 000009A4 3B0D[A0760100] cmp ecx, [k_page_dir]
1977 000009AA 7409 je short T3
1978 ;
1979 000009AC 8B0D[A0760100] mov ecx, [k_page_dir]
1980 000009B2 0F22D9 mov cr3, ecx
1981 T3:
1982 ;sti ; INTERRUPTS BACK ON
1983 000009B5 66FF05[20770100] INC word [TIMER_LOW] ; INCREMENT TIME
1984 000009BC 7507 JNZ short T4 ; GO TO TEST_DAY
1985 000009BE 66FF05[22770100] INC word [TIMER_HIGH] ; INCREMENT HIGH WORD OF TIME
1986 ; TEST_DAY
1987 000009C5 66833D[22770100]18 CMP word [TIMER_HIGH],018H ; TEST FOR COUNT EQUALING 24 HOURS
1988 000009CD 7519 JNZ short T5 ; GO TO DISKETTE_CTL
1989 000009CF 66813D[20770100]B0- CMP word [TIMER_LOW],0B0H
1990 000009D7 00
1991 000009D8 750E JNZ short T5 ; GO TO DISKETTE_CTL
1992 ;-----
1993 ;SUB AX,AX
1994 ;MOV [TIMER_HIGH],AX
1995 ;MOV [TIMER_LOW],AX
1996 000009DA 29C0 sub eax, eax
1997 000009DC A3[20770100] mov [TIMER_LH], eax
1998 ;
1999 000009E1 C605[24770100]01 MOV byte [TIMER_OFL],1
2000 ;-----
2001 ; TEST FOR DISKETTE TIME OUT
2002 ;
2003 T5:

```

```

2004 ; 23/12/2014
2005 000009E8 EB1D jmp short T6 ; will be replaced with nop, nop
2006 ; (9090h) if a floppy disk
2007 ; is detected.
2008 ;mov al,[CS:MOTOR_COUNT]
2009 mov al,[MOTOR_COUNT]
2010 000009EF FEC8 dec al
2011 ;mov [CS:MOTOR_COUNT],al ; DECREMENT DISKETTE MOTOR CONTROL
2012 000009F1 A2[27770100] mov [MOTOR_COUNT],al
2013 ;mov [ORG_MOTOR_COUNT],al
2014 000009F6 750F JNZ short T6 ; RETURN IF COUNT NOT OUT
2015 000009F8 B0F0 mov al,0F0h
2016 ;AND [CS:MOTOR_STATUS],al ; TURN OFF MOTOR RUNNING BITS
2017 000009FA 2005[26770100] and [MOTOR_STATUS],al
2018 ;and [ORG_MOTOR_STATUS],al
2019 00000A00 B00C MOV AL,0CH ; bit 3 = enable IRQ & DMA,
2020 ; bit 2 = enable controller
2021 ; 1 = normal operation
2022 ; 0 = reset
2023 ; bit 0, 1 = drive select
2024 ; bit 4-7 = motor running bits
2025 00000A02 66BAF203 MOV DX,03F2H ; FDC CTL PORT
2026 00000A06 EE OUT DX,AL ; TURN OFF THE MOTOR
2027
2028 T6: ;inc word [CS:wait_count] ; 22/12/2014 (byte -> word)
2029 ; TIMER TICK INTERRUPT
2030 ;inc word [wait_count] ; 27/02/2015
2031 ;INT 1CH ; TRANSFER CONTROL TO A USER ROUTINE
2032 ;cli
2033 00000A07 E85E040000 call u_timer ; TRANSFER CONTROL TO A USER ROUTINE
2034 ; 23/05/2016
2035 00000A0C E825110100 call clock ; Multi Tasking control procedure
2036
2037 T7: ; 14/10/2015
2038 00000A11 B020 MOV AL,E0I ; GET END OF INTERRUPT MASK
2039 00000A13 FA CLI ; DISABLE INTERRUPTS TILL STACK CLEARED
2040 00000A14 E620 OUT INTA00,AL ; END OF INTERRUPT TO 8259 - 1
2041 ;
2042 ;
2043 ; 21/08/2024
2044 00000A16 803D[31830100]FF cmp byte [p_change], 0FFh ; CTRL+BREAK signature
2045 00000A1D 7424 je short T9 ; current program will be forced to sysrelease
2046 ;
2047 rtc_int_2:
2048 ; 26/12/2016
2049 ;mov ecx,[cr3reg]
2050 ; 13/01/2017
2051 00000A1F 803D[CC8E0100]00 cmp byte [u.t_lock], 0 ; T_LOCK
2052 00000A26 7730 ja short timer_int_return ; Timer Lock : 'sysrele' is needed !
2053 ; 28/02/2017
2054 ; We need to exit if the user's IRQ callback service is in progress!
2055 ; (To prevent a conflict!)
2056 00000A28 803D[D08E0100]00 cmp byte [u.r_lock], 0 ; R_LOCK, IRQ callback service lock !
2057 00000A2F 7727 ja short timer_int_return ; Timer Lock : 'sysrele' is needed !
2058 ; 15/01/2017
2059 00000A31 803D[30830100]02 cmp byte [priority], 2
2060 00000A38 733A jnb short T8 ; current process has a timer event (15/01/2017)
2061 ; 22/05/2016
2062 00000A3A 803D[31830100]00 cmp byte [p_change], 0 ; in 'set_run_sequence', in 'rtc_p'
2063 00000A41 7615 jna short timer_int_return ; 23/05/2016
2064
2065 ; 15/01/2017
2066 ; present process must be changed with high priority process
2067 T9: ; 21/08/2024
2068 ;xor al,al
2069 00000A43 31C0 xor eax,eax ; 26/12/2016
2070 00000A45 A2[31830100] mov [p_change],al ; 0
2071 ;mov byte [priority], 2 ; 15/01/2017 (there is a timer event)
2072
2073 00000A4A 803D[488E0100]FF cmp byte [sysflg], 0FFh ; user or system space ?
2074 00000A51 7416 je short rtc_int_3 ; user space ([sysflg]= 0FFh)
2075
2076 ; system space, wait for 'sysret'
2077 ; to change running process
2078 ; with high priority (event) process
2079
2080 00000A53 A2[9C8E0100] mov [u.quant],al ; 0
2081
2082 timer_int_return: ; 23/05/2016 - jump from 'rtc_int' ('rtc_int_2')
2083 00000A58 8B0D[648F0100] mov ecx,[cr3reg] ; previous value/content of cr3 register
2084 00000A5E 0F22D9 mov cr3,ecx ; restore cr3 register content
2085 ;
2086 00000A61 61 popad ; edi, esi, ebp, temp (increment esp by 4), ebx, edx, ecx, eax
2087 ;
2088 00000A62 0FA9 pop gs
2089 00000A64 0FA1 pop fs
2090 00000A66 07 pop es
2091 00000A67 1F pop ds
2092 ;
2093 00000A68 CF iretd ; return from interrupt
2094
2095 rtc_int_3:
2096 00000A69 FE05[488E0100] inc byte [sysflg] ; now, we are in system space
2097 ;
2098 00000A6F E92BC00000 jmp sysrelease ; change running process immediately
2099
2100 T8:
2101 ; 13/01/2017 (eax -> ebx)
2102 ; callback checking... (19/12/2016)
2103 00000A74 31DB xor ebx,ebx
2104 00000A76 871D[C88E0100] xchg ebx,[u.tcb] ; callback address (0 = normal return)
2105 00000A7C 09DB or ebx,ebx
2106 00000A7E 74D8 jz short timer_int_return
2107
2108 ; Set user's callback routine as return address from this interrupt
2109 ; and set normal return address as return address from callback
2110 ; routine!!! (19/12/2016)
2111
2112 ; 14/01/2017
2113 ; 13/01/2017 - Timer Lock (T_LOCK)
2114 00000A80 FE05[CC8E0100] inc byte [u.t_lock]
2115 00000A86 8A0D[488E0100] mov cl,[sysflg]
2116 00000A8C 880D[CD8E0100] mov [u.t_mode],cl
2117
2118 00000A92 8B2D[3C760100] mov ebp,[tss.esp0] ; kernel stack address (for ring 0)
2119 00000A98 83ED14 sub ebp,20 ; eip, cs, eflags, esp, ss
2120 00000A9B 892D[4C8E0100] mov [u.sp],ebp
2121 00000AA1 8925[508E0100] mov [u.usp],esp
2122
2123 ;or word [ebp+8], 200h ; 22/01/2017, force enabling interrupts
2124
2125 00000AA7 8B44241C mov eax,[esp+28] ; pushed eax
2126 00000AAB A3[548E0100] mov [u.r0],eax
2127

```

```

2128 00000AB0 E812FF0000      call    wswap ; save user's registers & status
2129
2130      ; software int is in ring 0 but timer int must return to ring 3
2131      ; so, ring 3 return address and stack registers
2132      ; (eip, cs, eflags, esp, ss)
2133      ; must be copied to timer int return
2134      ; eip will be replaced by callback service routine address
2135
2136 00000AB5 C605[488E0100]FF  mov     byte [sysflg], 0FFh ; user mode
2137
2138      ; system mode (system call)
2139      ;mov     ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),
2140      ;         ; ESP (u), SS (UDATA)
2141
2142 00000ABC 8B4510      mov     eax, [ebp+16] ; SS (UDATA)
2143 00000ABF 89E6      mov     esi, esp
2144 00000AC1 50      push    eax
2145 00000AC2 50      push    eax
2146 00000AC3 89E7      mov     edi, esp
2147 00000AC5 893D[508E0100]  mov     [u.usp], edi
2148 00000ACB B908000000      mov     ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS
2149 00000AD0 F3A5      rep     movsd
2150 00000AD2 B104      mov     cl, 4
2151 00000AD4 F3AB      rep     stosd
2152 00000AD6 893D[4C8E0100]  mov     [u.sp], edi
2153 00000ADC 89EE      mov     esi, ebp
2154 00000ADE B105      mov     cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)
2155 00000AE0 F3A5      rep     movsd
2156
2157 00000AE2 8B0D[AC8E0100]  mov     ecx, [u.pgdir]
2158 00000AE8 890D[648F0100]  mov     [cr3reg], ecx
2159
2160      ; 13/01/2017 (eax -> ebx)
2161      ; EBX = callback routine address (virtual, not physical address!)
2162
2163      ; 09/01/2017
2164      ; !!! CALLBACK ROUTINE MUST BE ENDED/RETURNED WITH 'sysrele'
2165      ; system call !!!
2166      ; 25/12/2016
2167      ; callback Note: (19/12/2016)
2168      ; !!! CALLBACK ROUTINE MUST BE ENDED/RETURNED WITH 'RETN' !!!
2169      ; pushf ; save flags
2170      ; <callback service code>
2171      ; popf ; restore flags
2172      ; retn ; return to normal running address
2173
2174
2175      ; 15/01/2017
2176      ; 14/01/2017
2177      ; 13/01/2017 (eax -> ebx)
2178      ; 10/01/2017
2179      set_callback_addr:
2180      ; 09/01/2017 (**)
2181      ; 02/01/2017 (*)
2182      ; 25/12/2016 (*)
2183      ; 19/12/2016 (TRDOS 386 feature only!)
2184
2185      ; This routine sets return address
2186      ; to start of user's interrupt
2187      ; service (callback) address
2188      ; and sets callback 'retn' address to normal
2189      ; return address of user's running code!
2190
2191      ; INPUT:
2192      ; EBX = callback routine/service address
2193      ;         (virtual, not physical address!)
2194      ; [u.sp] = kernel stack, points to
2195      ;         user's EIP,CS,EFLAGS,ESP,SS
2196      ;         registers.
2197      ; OUTPUT:
2198      ; EIP (user) = callback (service) address
2199      ; CS (user) = UCODE
2200      ; EFLAGS (user) = flags before callback
2201      ; ESP (user) = ESP-4 (user, before callback)
2202      ; [ESP](user) = EIP (user) before callback
2203
2204      ; Note: If CPU was in user mode while entering
2205      ; the timer interrupt service routine,
2206      ; 'IRET' will get return to callback routine
2207      ; immediately. If CPU was in system/kernel mode
2208      ; 'iret' will get return to system call and
2209      ; then, callback routine will be return address
2210      ; from system call. (User's callback/service code
2211      ; will be able to return to normal return address
2212      ; via an 'retn' at the end.)
2213
2214      ; Note(**): User's callback service code must be ended
2215      ; with a 'sysrele' system call ! (09/01/2017)
2216
2217      ; For example:
2218
2219      ; timer_callback:
2220      ;     inc dword [time_counter]
2221      ;     mov eax, 39 ; 'sysrele'
2222      ;     int 40h ; TRDOS 386 system call (interrupt)
2223
2224
2225      ; Note(*): User's callback service code must preserve cpu
2226      ; flags if it has any instructions which changes
2227      ; flags in the service code. (25/12/2016)
2228
2229      ; For example:
2230
2231      ; timer_callback:
2232      ;     pushf ; save flags
2233      ;     ; this instruction changes zero flag
2234      ;     inc dword [time_counter]
2235      ;     popf ; restore flags
2236      ;     retn ; return to normal user code
2237      ;         (which is interrupted by the
2238      ;         timer interput)
2239
2240
2241
2242
2243      ; 15/01/2017
2244 00000AEE 8B2D[4C8E0100]  mov     ebp, [u.sp]; kernel's stack, points to EIP (user)
2245 00000AF4 895D00      mov     [ebp], ebx
2246 00000AF7 E95CFFFFFF      jmp     timer_int_return
2247
2248      ; 15/01/2017
2249      ; 13/01/2017
2250      ; 19/12/2016
2251      ; 06/06/2016

```

```

2252 ; 23/05/2016
2253 ; 22/05/2016
2254 ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
2255 ; 26/02/2015
2256 ; 07/09/2014
2257 ; 25/08/2014
2258 rtc_int: ; Real Time Clock Interrupt (IRQ 8)
2259 ; 22/05/2016
2260 00000AFC 1E push ds ; ** ; 23/05/2016
2261 00000AFD 50 push eax ; *
2262 00000AFE 66B81000 mov ax, KDATA
2263 00000B02 8ED8 mov ds, ax
2264 ;
2265 00000B04 8A25[1E770100] mov ah, [RTC_2Hz] ; 2 Hz interrupt to 1 Hz function
2266 00000B0A 80F401 xor ah, 1
2267 00000B0D 8825[1E770100] mov [RTC_2Hz], ah ; 1 = 0.5 second, 0 = 1 second
2268 00000B13 753B jnz short rtc_int_return ; half second
2269 ; 1 second
2270 rtc_int_0:
2271 ; 22/05/2016
2272 00000B15 58 pop eax ; *
2273 ;
2274 ; 14/10/2015 ('timer_int')
2275 ; Here, we are simulating system call entry (for task switch)
2276 ; (If multitasking is enabled,
2277 ; 'clock' procedure may jump to 'sysrelease')
2278 ; push ds ; ** ; 23/05/2016
2279 00000B16 06 push es
2280 00000B17 0FA0 push fs
2281 00000B19 0FA8 push gs
2282 00000B1B 60 pushad ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
2283 00000B1C 66B91000 mov cx, KDATA
2284 ; mov ds, cx ; 06/06/2016
2285 00000B20 8EC1 mov es, cx
2286 00000B22 8EE1 mov fs, cx
2287 00000B24 8EE9 mov gs, cx
2288 ;
2289 00000B26 0F20D9 mov ecx, cr3
2290 00000B29 890D[648F0100] mov [cr3reg], ecx ; save current cr3 register value/content
2291 ;
2292 00000B2F 803D[CC8E0100]00 cmp byte [u.t_lock], 0 ; timer lock (callback) status ?
2293 00000B36 7711 ja short rtc_int_1 ; yes
2294 ;
2295 ; 15/01/2017
2296 00000B38 3B0D[A0760100] cmp ecx, [k_page_dir]
2297 00000B3E 7409 je short rtc_int_1
2298 ;
2299 00000B40 8B0D[A0760100] mov ecx, [k_page_dir]
2300 00000B46 0F22D9 mov cr3, ecx
2301 rtc_int_1:
2302 ; Timer event (kernel) functions must be performed with
2303 ; 1 second intervals - TRDOS 386 (TRDOS v2.0) feature ! -
2304 ;
2305 ; 25/08/2014
2306 00000B49 E818030000 call rtc_p ; 19/05/2016 - major modification
2307 ;
2308 ; 23/05/2016
2309 00000B4E 28E4 sub ah, ah ; 0
2310 ; 22/05/2016 - TRDOS 386 timer event modifications
2311 rtc_int_return: ; 19/05/2016
2312 ; 22/02/2015 - dsectpm.s
2313 ; [ source: http://wiki.osdev.org/RTC ]
2314 ; read status register C to complete procedure
2315 ; (it is needed to get a next IRQ 8)
2316 00000B50 B00C mov al, 0Ch ;
2317 00000B52 E670 out 70h, al ; select register C
2318 00000B54 90 nop
2319 00000B55 E471 in al, 71h ; just throw away contents
2320 ; 22/02/2015
2321 00000B57 B020 MOV AL,EOI ; END OF INTERRUPT
2322 ; CLI ; DISABLE INTERRUPTS TILL STACK CLEARED
2323 00000B59 E6A0 OUT INTB00,AL ; FOR CONTROLLER #2
2324 ;
2325 ; 23/05/2016
2326 00000B5B B020 MOV AL,EOI ; GET END OF INTERRUPT MASK
2327 00000B5D FA CLI ; DISABLE INTERRUPTS TILL STACK CLEARED
2328 00000B5E E620 OUT INTA00,AL ; END OF INTERRUPT TO 8259 - 1
2329 ;
2330 ; 23/05/2016
2331 00000B60 20E4 and ah, ah
2332 ; jz rtc_int_2
2333 ; 24/07/2022
2334 00000B62 7505 jnz short rtc_int_4
2335 00000B64 E9B6FEFFFF jmp rtc_int_2
2336 rtc_int_4:
2337 ; ah = 1 (half second)
2338 00000B69 58 pop eax ; *
2339 00000B6A 1F pop ds ; **
2340 00000B6B CF iretd
2341 ;
2342 ; //////////////////////////////////
2343 ; 28/08/2014
2344 ;
2345 irq0:
2346 00000B6C 6A00 push dword 0
2347 00000B6E EB48 jmp short which_irq
2348 irq1:
2349 00000B70 6A01 push dword 1
2350 00000B72 EB44 jmp short which_irq
2351 irq2:
2352 00000B74 6A02 push dword 2
2353 00000B76 EB40 jmp short which_irq
2354 irq3:
2355 ; 20/11/2015
2356 ; 24/10/2015
2357 00000B78 2EFF15[C01C0100] call dword [cs:com2_irq3]
2358 00000B7F 6A03 push dword 3
2359 00000B81 EB35 jmp short which_irq
2360 irq4:
2361 ; 20/11/2015
2362 ; 24/10/2015
2363 00000B83 2EFF15[BC1C0100] call dword [cs:com1_irq4]
2364 00000B8A 6A04 push dword 4
2365 00000B8C EB2A jmp short which_irq
2366 irq5:
2367 00000B8E 6A05 push dword 5
2368 00000B90 EB26 jmp short which_irq
2369 irq6:
2370 00000B92 6A06 push dword 6
2371 00000B94 EB22 jmp short which_irq
2372 irq7:
2373 00000B96 6A07 push dword 7
2374 00000B98 EB1E jmp short which_irq
2375 irq8:

```



```

2376 00000B9A 6A08      push     dword 8
2377 00000B9C EB1A      jmp      short which_irq
2378
2379 00000B9E 6A09      push     dword 9
2380 00000BA0 EB16      jmp      short which_irq
2381
2382 00000BA2 6A0A      push     dword 10
2383 00000BA4 EB12      jmp      short which_irq
2384
2385 00000BA6 6A0B      push     dword 11
2386 00000BA8 EB0E      jmp      short which_irq
2387
2388 00000BAA 6A0C      push     dword 12
2389 00000BAC EB0A      jmp      short which_irq
2390
2391 00000BAE 6A0D      push     dword 13
2392 00000BB0 EB06      jmp      short which_irq
2393
2394 00000BB2 6A0E      push     dword 14
2395 00000BB4 EB02      jmp      short which_irq
2396
2397 00000BB6 6A0F      push     dword 15
2398      ;jmp     short which_irq
2399
2400      ; 22/01/2017
2401      ; 19/10/2015
2402      ; 29/08/2014
2403      ; 21/08/2014
2404
2405 00000BB8 870424    which_irq: xchg     eax, [esp] ; 28/08/2014
2406 00000BBB 53      push     ebx
2407 00000BBC 56      push     esi
2408 00000BBD 57      push     edi
2409 00000BBE 1E      push     ds
2410 00000BBF 06      push     es
2411      ;
2412 00000BC0 88C3      mov     bl, al
2413      ;
2414 00000BC2 B810000000    mov     eax, KDATA
2415 00000BC7 8ED8      mov     ds, ax
2416 00000BC9 8EC0      mov     es, ax
2417      ; 19/10/2015
2418 00000BCB FC      cld
2419      ; 27/08/2014
2420 00000BCC 8105[E0350100]A000-    add     dword [scr_row], 0A0h
2421 00000BD4 0000
2422
2422 00000BD6 B417      mov     ah, 17h ; blue (1) background,
2423      ; light gray (7) forecolor
2424 00000BD8 8B3D[E0350100]      mov     edi, [scr_row]
2425 00000BDE B049      mov     al, 'I'
2426 00000BE0 66AB      stosw
2427 00000BE2 B052      mov     al, 'R'
2428 00000BE4 66AB      stosw
2429 00000BE6 B051      mov     al, 'Q'
2430 00000BE8 66AB      stosw
2431 00000BEA B020      mov     al, ' '
2432 00000BEC 66AB      stosw
2433 00000BEE 88D8      mov     al, bl
2434 00000BF0 3C0A      cmp     al, 10
2435 00000BF2 7208      jb      short ii1
2436 00000BF4 B031      mov     al, 'l'
2437 00000BF6 66AB      stosw
2438 00000BF8 88D8      mov     al, bl
2439 00000BFA 2C0A      sub     al, 10
2440
2441 00000BFC 0430      ii1:      add     al, '0'
2442 00000BFE 66AB      stosw
2443 00000C00 B020      mov     al, ' '
2444 00000C02 66AB      stosw
2445 00000C04 B021      mov     al, '!'
2446 00000C06 66AB      stosw
2447 00000C08 B020      mov     al, ' '
2448 00000C0A 66AB      stosw
2449      ; 23/02/2015
2450 00000C0C 80FB07      cmp     bl, 7 ; check for IRQ 8 to IRQ 15
2451 00000C0F 7604      jna     ii2
2452      ; 22/01/2017
2453 00000C11 B020      mov     al, 20h ; END OF INTERRUPT COMMAND TO
2454 00000C13 E6A0      out     0A0h, al ; the 2nd 8259
2455
2456 00000C15 B020      ii2:      mov     al, 20h ; END OF INTERRUPT COMMAND TO
2457 00000C17 E620      out     20h, al ; the 2nd 8259
2458 00000C19 E9CA010000    jmp     iiret
2459      ;
2460      ; 22/08/2014
2461      ;mov     al, 20h ; END OF INTERRUPT COMMAND TO 8259
2462      ;out     20h, al ; 8259 PORT
2463      ;
2464      ;pop     es
2465      ;pop     ds
2466      ;pop     edi
2467      ;pop     esi
2468      ;pop     ebx
2469      ;pop     eax
2470      ;iret
2471
2472      ; 02/04/2015
2473      ; 25/08/2014
2474
2475 00000C1E 6A00      exc0:      push     dword 0
2476 00000C20 E990000000    jmp     cpu_except
2477
2478 00000C25 6A01      exc1:      push     dword 1
2479 00000C27 E989000000    jmp     cpu_except
2480
2481 00000C2C 6A02      exc2:      push     dword 2
2482 00000C2E E982000000    jmp     cpu_except
2483
2484 00000C33 6A03      exc3:      push     dword 3
2485 00000C35 EB7E      jmp     cpu_except
2486
2487 00000C37 6A04      exc4:      push     dword 4
2488 00000C39 EB7A      jmp     cpu_except
2489
2490 00000C3B 6A05      exc5:      push     dword 5
2491 00000C3D EB76      jmp     cpu_except
2492
2493 00000C3F 6A06      exc6:      push     dword 6
2494 00000C41 EB72      jmp     cpu_except
2495
2496 00000C43 6A07      exc7:      push     dword 7
2497 00000C45 EB6E      jmp     cpu_except
2498
2498

```

```

2499         ; [esp] = Error code
2500         push    dword 8
2501         jmp     cpu_except_en
2502
2503         push    dword 9
2504         jmp     cpu_except
2505
2506         ; [esp] = Error code
2507         push    dword 10
2508         jmp     cpu_except_en
2509
2510         ; [esp] = Error code
2511         push    dword 11
2512         jmp     cpu_except_en
2513
2514         ; [esp] = Error code
2515         push    dword 12
2516         jmp     cpu_except_en
2517
2518         ; [esp] = Error code
2519         push    dword 13
2520         jmp     cpu_except_en
2521
2522         ; [esp] = Error code
2523         push    dword 14
2524         jmp     short cpu_except_en
2525
2526         push    dword 15
2527         jmp     cpu_except
2528
2529         push    dword 16
2530         jmp     cpu_except
2531
2532         ; [esp] = Error code
2533         push    dword 17
2534         jmp     short cpu_except_en
2535
2536         push    dword 18
2537         jmp     short cpu_except
2538
2539         push    dword 19
2540         jmp     short cpu_except
2541
2542         push    dword 20
2543         jmp     short cpu_except
2544
2545         push    dword 21
2546         jmp     short cpu_except
2547
2548         push    dword 22
2549         jmp     short cpu_except
2550
2551         push    dword 23
2552         jmp     short cpu_except
2553
2554         push    dword 24
2555         jmp     short cpu_except
2556
2557         push    dword 25
2558         jmp     short cpu_except
2559
2560         push    dword 26
2561         jmp     short cpu_except
2562
2563         push    dword 27
2564         jmp     short cpu_except
2565
2566         push    dword 28
2567         jmp     short cpu_except
2568
2569         push    dword 29
2570         jmp     short cpu_except
2571
2572         push    dword 30
2573         jmp     short cpu_except_en
2574
2575         push    dword 31
2576         jmp     short cpu_except
2577
2578         ; 19/10/2015
2579         ; 19/09/2015
2580         ; 01/09/2015
2581         ; 28/08/2015
2582         ; 28/08/2014
2583
2584         xchg    eax, [esp+4] ; Error code
2585         mov     [ss:error_code], eax
2586         pop     eax ; Exception number
2587         xchg    eax, [esp]
2588         ; eax = eax before exception
2589         ; [esp] -> exception number
2590         ; [esp+4] -> EIP to return
2591
2592         ; 22/01/2017
2593         ; 19/10/2015
2594         ; 19/09/2015
2595         ; 01/09/2015
2596         ; 28/08/2015
2597         ; 29/08/2014
2598         ; 28/08/2014
2599         ; 25/08/2014
2600         ; 21/08/2014
2601
2602         cpu_except: ; CPU Exceptions
2603         cld
2604         xchg    eax, [esp]
2605         ; eax = Exception number
2606         ; [esp] = eax (before exception)
2607
2608         push    ebx
2609         push    esi
2610         push    edi
2611         push    ds
2612         push    es
2613         ; 28/08/2015
2614         mov     bx, KDATA
2615         mov     ds, bx
2616         mov     es, bx
2617         mov     ebx, cr3
2618         push    ebx ; (*) page directory
2619         ; 19/10/2015
2620         cld
2621         ; 25/03/2015
2622         mov     ebx, [k_page_dir]
2623         mov     cr3, ebx
2624         ; 28/08/2015
2625         cmp     eax, 0Eh ; 14, PAGE FAULT

```

```

2623 00000CD7 7510          jne     short cpu_except_nfp
2624 00000CD9 E8EF4D0000    call    page_fault_handler
2625 00000CDE 21C0          and     eax, eax
2626                                     ;jz     iretp ; 01/09/2015
2627                                     ; 24/07/2022
2628 00000CE0 7505          jnz     short cpu_except_pf
2629 00000CE2 E9FD000000    jmp     iretp
2630 cpu_except_pf:           ; 24/07/2022
2631 mov     al, 0Eh ; 14
2632 cpu_except_nfp:
2633                                     ; 23/08/2016
2634 00000CE9 803D[1E680000]03    cmp     byte [CRT_MODE], 3
2635 00000CF0 7409          je      short cpu_except_mode_3
2636 00000CF2 50          push    eax
2637 00000CF3 B003          mov     al, 3
2638 00000CF5 E88F0E0000    call    _set_mode
2639 00000CFA 58          pop     eax
2640 cpu_except_mode_3:
2641                                     ; 02/04/2015
2642 00000CFB BB[FD080000]    mov     ebx, hang
2643 00000D00 875C241C    xchg    ebx, [esp+28]
2644                                     ; EIP (points to instruction which faults)
2645                                     ; New EIP (hang)
2646 00000D04 891D[6C900100]    mov     [FaultOffset], ebx
2647 00000D0A C744242008000000    mov     dword [esp+32], KCODE ; kernel's code segment
2648 00000D12 814C242400020000    or      dword [esp+36], 200h ; enable interrupts (set IF)
2649                                     ;
2650 00000D1A 88C4          mov     ah, al
2651 00000D1C 240F          and     al, 0Fh
2652 00000D1E 3C09          cmp     al, 9
2653 00000D20 7602          jna     short h1ok
2654 00000D22 0407          add     al, 'A'-' ':'
2655 h1ok:
2656 00000D24 C0EC04          shr     ah, 4
2657 00000D27 80FC09          cmp     ah, 9
2658 00000D2A 7603          jna     short h2ok
2659 00000D2C 80C407          add     ah, 'A'-' ':'
2660 h2ok:
2661 00000D2F 86C4          xchg    ah, al
2662 00000D31 66053030    add     ax, '00'
2663 00000D35 66A3[38380100]    mov     [excnsr], ax
2664                                     ;
2665                                     ; 29/08/2014
2666 00000D3B A1[6C900100]    mov     eax, [FaultOffset]
2667 00000D40 51          push    ecx
2668 00000D41 52          push    edx
2669 00000D42 89E3          mov     ebx, esp
2670                                     ; 28/08/2015
2671 00000D44 B910000000    mov     ecx, 16 ; divisor value to convert binary number
2672                                     ; to hexadecimal string
2673                                     ; divisor to convert
2674                                     ; binary number to decimal string
2675 b2d1:
2676 00000D49 31D2          xor     edx, edx
2677 00000D4B F7F1          div     ecx
2678                                     ;push    dx
2679                                     ; 18/04/2021
2680 00000D4D 52          push    edx
2681 00000D4E 39C8          cmp     eax, ecx
2682 00000D50 73F7          jnb     short b2d1
2683 00000D52 BF[43380100]    mov     edi, EIPstr ; EIP value
2684                                     ; points to instruction which faults
2685                                     ; 28/08/2015
2686 00000D57 89C2          mov     edx, eax
2687 b2d2:
2688                                     ;add     al, '0'
2689 00000D59 8A82[09420000]    mov     al, [edx+hexchrs]
2690 00000D5F AA          stosb   ; write hexadecimal digit to its place
2691 00000D60 39E3          cmp     ebx, esp
2692 00000D62 7605          jna     short b2d3
2693                                     ;pop     ax
2694                                     ; 18/04/2021
2695 00000D64 58          pop     eax
2696 00000D65 88C2          mov     dl, al
2697 00000D67 EBF0          jmp     short b2d2
2698 b2d3:
2699 00000D69 B068          mov     al, 'h' ; 28/08/2015
2700 00000D6B AA          stosb
2701 00000D6C B020          mov     al, 20h ; space
2702 00000D6E AA          stosb
2703 00000D6F 30C0          xor     al, al ; to do it an ASCIIZ string
2704 00000D71 AA          stosb
2705                                     ;
2706 00000D72 5A          pop     edx
2707 00000D73 59          pop     ecx
2708                                     ;
2709 00000D74 B44F          mov     ah, 4Fh ; red (4) background,
2710                                     ; white (F) forecolor
2711 00000D76 BE[28380100]    mov     esi, exc_msg ; message offset
2712                                     ;
2713                                     ; 20/01/2017 (!cpu exception!)
2714                                     ;
2715 00000D7B 8105[E0350100]A000-    add     dword [scr_row], 0A0h
2716 00000D83 0000          mov     [scr_row], 0
2717 00000D85 8B3D[E0350100]    mov     edi, [scr_row]
2718 00000D8B C605[488E0100]00    mov     byte [sysflg], 0 ; system mode
2719 00000D92 FB          sti
2720                                     ;
2721 00000D93 E89FFBFFFF    call    printk
2722                                     ;
2723 00000D98 B410          mov     ah, 10h
2724 00000D9A E87D010000    call    int16h ; getc
2725                                     ;
2726 00000D9F B003          mov     al, 3
2727 00000DA1 E8E30D0000    call    _set_mode
2728                                     ;
2729                                     ;mov     eax, 1
2730                                     ; 30/11/2023
2731 00000DA6 31C0          xor     eax, eax
2732 00000DA8 40          inc     eax
2733                                     ; eax = 1
2734 00000DA9 E90EBE0000    jmp     sysexit ; terminate process !!!
2735                                     ;
2736                                     ; 22/01/2017
2737                                     ; 18/04/2016
2738                                     ; 28/08/2015
2739                                     ; 23/02/2015
2740                                     ; 20/08/2014
2741 ignore_int:
2742 00000DAE 50          push    eax
2743 00000DAF 53          push    ebx ; 23/02/2015
2744 00000DB0 56          push    esi
2745 00000DB1 57          push    edi

```

```

2746 00000DB2 1E      push    ds
2747 00000DB3 06      push    es
2748                ; 18/04/2016
2749 00000DB4 66B81000    mov     ax, KDATA
2750 00000DB8 8ED8      mov     ds, ax
2751 00000DBA 8EC0      mov     es, ax
2752                ; 28/08/2015
2753 00000DBC 0F20D8    mov     eax, cr3
2754 00000DBF 50      push    eax ; (*) page directory
2755                ;
2756 00000DC0 B467      mov     ah, 67h ; brown (6) background,
2757                ; light gray (7) forecolor
2758 00000DC2 BE[F0360100]    mov     esi, int_msg ; message offset
2759                piemsg:
2760                ; 27/08/2014
2761 00000DC7 8105[E0350100]A000-    add     dword [scr_row], 0A0h
2761 00000DCF 0000
2762 00000DD1 8B3D[E0350100]    mov     edi, [scr_row]
2763                ;
2764 00000DD7 E85BF8FFFF    call    printk
2765                ;
2766                ; 23/02/2015
2767 00000DDC B020      mov     al, 20h ; END OF INTERRUPT COMMAND TO
2768 00000DDE E6A0      out     0A0h, al ; the 2nd 8259
2769                ; 22/08/2014
2770 00000DE0 B020      mov     al, 20h ; END OF INTERRUPT COMMAND TO 8259
2771 00000DE2 E620      out     20h, al ; 8259 PORT
2772                iiretp:
2773                ; 22/01/2017
2774                ; 01/09/2015
2775                ; 28/08/2015
2776 00000DE4 58      pop     eax ; (*) page directory
2777 00000DE5 0F22D8    mov     cr3, eax
2778                iiret:
2779                pop     es
2780                pop     ds
2781                pop     edi
2782                pop     esi
2783                pop     ebx ; 29/08/2014
2784                pop     eax
2785                iretd
2786
2787                ; 23/05/2016
2788                ; 22/08/2014
2789                ; IBM PC/AT BIOS source code ----- 10/06/85 (bios.asm)
2790                ; (INT 1Ah)
2791                ; Linux (v0.12) source code (main.c) by Linus Torvalds (1991)
2792                time_of_day:
2793 00000DEF E829570000    call    UPD_IPR                ; WAIT TILL UPDATE NOT IN PROGRESS
2794 00000DF4 726F      jc     short time_of_day_retn ; 23/05/2016
2795 00000DF6 B000      mov     al, CMOS_SECONDS
2796 00000DF8 E856570000    call    CMOS_READ
2797 00000DFD A2[10770100]    mov     [time_seconds], al
2798 00000E02 B002      mov     al, CMOS_MINUTES
2799 00000E04 E84A570000    call    CMOS_READ
2800 00000E09 A2[11770100]    mov     [time_minutes], al
2801 00000E0E B004      mov     al, CMOS_HOURS
2802 00000E10 E83E570000    call    CMOS_READ
2803 00000E15 A2[12770100]    mov     [time_hours], al
2804 00000E1A B006      mov     al, CMOS_DAY_WEEK
2805 00000E1C E832570000    call    CMOS_READ
2806 00000E21 A2[13770100]    mov     [date_wday], al
2807 00000E26 B007      mov     al, CMOS_DAY_MONTH
2808 00000E28 E826570000    call    CMOS_READ
2809 00000E2D A2[14770100]    mov     [date_day], al
2810 00000E32 B008      mov     al, CMOS_MONTH
2811 00000E34 E81A570000    call    CMOS_READ
2812 00000E39 A2[15770100]    mov     [date_month], al
2813 00000E3E B009      mov     al, CMOS_YEAR
2814 00000E40 E80E570000    call    CMOS_READ
2815 00000E45 A2[16770100]    mov     [date_year], al
2816 00000E4A B032      mov     al, CMOS_CENTURY
2817 00000E4C E802570000    call    CMOS_READ
2818 00000E51 A2[17770100]    mov     [date_century], al
2819                ;
2820 00000E56 B000      mov     al, CMOS_SECONDS
2821 00000E58 E8F6560000    call    CMOS_READ
2822 00000E5D 3A05[10770100]    cmp     al, [time_seconds]
2823 00000E63 758A      jne     short time_of_day
2824
2825                time_of_day_retn:
2826 00000E65 C3      retn
2827
2828                ; 15/01/2017
2829                ; 10/06/2016
2830                ; 07/06/2016
2831                ; 06/06/2016
2832                ; 23/05/2016
2833                rtc_p:
2834 00000E66 B101      mov     cl, 1 ; 15/01/2017
2835 00000E68 EB02      jmp     short rtc_p0
2836                u_timer:
2837                ; Timer Events with 18.2 Hz Timer Ticks
2838                ; (and also timer events with RTC seconds)
2839 00000E6A 28C9      sub     cl, cl ; mov cl, 0 ; 15/01/2017
2840                rtc_p0:
2841                ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
2842                ; Major Modification:
2843                ; Check and Perform Timer Events (for RTC)
2844                ; 25/08/2014 - 07/09/2014
2845                ; Retro UNIX 386 v1:
2846                ; Print Real Time Clock content
2847                ;
2848                ; 15/01/2017
2849 00000E6C 880D[30830100]    mov     byte [priority], cl ; 0 or 1 (not 2)
2850 00000E72 8A2D[33830100]    mov     ch, [timer_events]
2851 00000E78 20ED      and     ch, ch
2852 00000E7A 7420      jz     short rtc_p3
2853
2854 00000E7C BE[688F0100]    mov     esi, timer_set ; beginning address of
2855                ; timer events space
2856                rtc_p1:
2857                mov     eax, [esi]
2858 00000E83 20C0      and     al, al ; 0 = free, >0 = process no.
2859 00000E85 7416      jz     short rtc_p4
2860                ;
2861 00000E87 C1C810      ror     eax, 16
2862                ; ah = response value, al = interrupt type
2863                ; 15/01/2017
2864                ; cl = interrupt source
2865                ; 1 = RTC, 0 = PIT
2866 00000E8A 38C8      cmp     al, cl
2867 00000E8C 750A      jne     short rtc_p2 ; not as requested or undefined !
2868 00000E8E 3C01      cmp     al, 1 ; 1 ; RTC interrupt ?

```

```

2869 00000E90 7410          je      short rtc_p5 ; yes, check for response
2870                          ; 06/06/2016 - 18.2 Hz Timer Ticks
2871 00000E92 836E080A      sub     dword [esi+8], 10 ; 1 tick = 10
2872 00000E96 7613          jna     short rtc_p6 ; continue for responding
2873                          rtc_p2:
2874                          ; 15/01/2017 (c1 -> ch)
2875                          ; 07/06/2016
2876 00000E98 FECD          dec     ch ; remain count of timer events
2877 00000E9A 7501          jnz     short rtc_p4
2878                          rtc_p3:
2879 00000E9C C3             retn
2880                          rtc_p4:
2881                          ; cmp     esi, timer_set + 240 ; 15*16 (last event)
2882                          ; jnb     short rtc_p3 ; end of timer event space
2883 00000E9D 83C610          add     esi, 16 ; next timer event
2884 00000EA0 EBD F          jmp     short rtc_p1
2885                          rtc_p5:
2886                          ; current timer count ; 06/06/2016 (182)
2887 00000EA2 816E08B6000000 sub     dword [esi+8], 182 ; 1 second (10*18.2)
2888 00000EA9 77ED          ja      short rtc_p2 ; check for the next
2889                          rtc_p6:
2890                          ; it is the time of response!
2891 00000EAB 8B5E04          mov     ebx, [esi+4] ; set (count limit) value
2892 00000EAE 895E08          mov     [esi+8], ebx ; reset count down value
2893                          ; to count limit
2894                          ; 19/12/2016
2895                          ; 10/12/2016 - timer callback modification
2896 00000EB1 8B7E0C          mov     edi, [esi+12] ; response (or callback) address
2897 00000EB4 807E0100      cmp     byte [esi+1], 0 ; >0 = callback
2898 00000EB8 762A          jna     short rtc_p8
2899                          ; timer callback !
2900                          movzx    ebx, byte [esi] ; process number (>0)
2901 00000EBA 0FB61E          mov     eax, ebx
2902 00000EBD 89D8          shl     bl, 2 ; *4
2903 00000EBF C0E302          mov     [ebx+p.tcb-4], edi ; user's callback service addr
2904 00000EC2 89BB[E48D0100]  cmp     al, [u.uno]
2905 00000EC8 3A05[A58E0100]  jne     short rtc_p9
2906 00000ECE 7521          mov     [u.tcb], edi
2907 00000ED0 893D[C88E0100]
2908                          rtc_p7:
2909                          ; 15/01/2017
2910 00000ED6 B002          mov     al, 2
2911 00000ED8 A2[30830100]      mov     [priority], al ; 2
2912                          ; 10/01/2017
2913                          ; mov     byte [u.pri], 2
2914 00000EDD A2[9E8E0100]      mov     [u.pri], al ; 2
2915 00000EE2 EBB4          jmp     short rtc_p2
2916                          rtc_p8:
2917                          ; response address is physical address of
2918                          ; the program's response (signal return) byte
2919                          ; 06/06/2016
2920                          ; mov     edi, [esi+12] ; response address
2921 00000EE4 8827          mov     [edi], ah ; response value
2922                          ;
2923 00000EE6 C1C010      rol     eax, 16
2924                          ; 15/01/2017
2925 00000EE9 3A05[A58E0100]  cmp     al, [u.uno] ; running process ?
2926 00000EEF 74E5          je      short rtc_p7
2927                          rtc_p9:
2928                          ; al = process number ; 10/06/2016
2929 00000EF1 B202          mov     dl, 2 ; priority, 2 = event (high)
2930 00000EF3 E8F30B0100    call    set_run_sequence ; 19/05/2016
2931 00000EF8 EB9E          jmp     short rtc_p2 ; 10/06/2016
2932
2933                          ; Default IRQ 7 handler against spurious IRQs (from master PIC)
2934                          ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
2935                          default_irq7:
2936                          ; push     ax
2937                          ; 18/04/2021
2938 00000EFA 50             push    eax
2939 00000EFB B00B          mov     al, 0Bh ; In-Service register
2940 00000EFD E620          out     20h, al
2941 00000EFF E800          jmp     short $+2
2942 00000F01 E800          jmp     short $+2
2943 00000F03 E420          in      al, 20h
2944 00000F05 2480          and     al, 80h ; bit 7 (is it real IRQ 7 or fake?)
2945 00000F07 7404          jz      short irq7_iret ; Fake (spurious) IRQ, do not send EOI
2946 00000F09 B020          mov     al, 20h ; EOI
2947 00000F0B E620          out     20h, al
2948                          irq7_iret:
2949                          ; pop      ax
2950                          ; 18/04/2021
2951 00000F0D 58             pop     eax
2952 00000F0E CF          iretd
2953
2954                          bcd_to_ascii:
2955                          ; 25/08/2014
2956                          ; INPUT ->
2957                          ; al = Packed BCD number
2958                          ; OUTPUT ->
2959                          ; ax = ASCII word/number
2960                          ;
2961                          ; Erdogan Tan - 1998 (proc_hex) - TRDOS.ASM (2004-2011)
2962                          ;
2963 00000F0F D410          db      0D4h, 10h ; Undocumented inst. AAM
2964                          ; AH = AL / 10h
2965                          ; AL = AL MOD 10h
2966 00000F11 660D3030      or      ax, '00' ; Make it ASCII based
2967
2968 00000F15 86C4          xchg    ah, al
2969
2970 00000F17 C3             retn
2971
2972                          ; 15/12/2020
2973                          real_mem_16m_64k:
2974 00000F18 0000          dw      0 ; Real size of system memory (if > 16MB)
2975                          ; as number of 64K blocks - 256
2976                          ; (This is for saving real system memory
2977                          ; because if system memory is larger than
2978                          ; 3 GB and if a VESA VBE video bios
2979                          ; is detected, 'mem_16m_64k' may be
2980                          ; decreased to reserve LFB space
2981                          ; at the end of system memory.)
2982                          ; Upper memory space from LFB base address
2983                          ; to 4GB will not be included by M.A.T.
2984                          def_LFB_addr:
2985 00000F1A 0000          dw      0 ; HW of default LFB addr (for mode 118h)
2986
2987
2988                          %include 'keyboard.s' ; 07/03/2015
2989                          <1> ; *****
2990                          <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.5 - keyboard.s
2991                          <1> ; -----
2992                          <1> ; Last update: 07/08/2022 (Previous: 12/04/2021)

```

```

5      <1> ; -----
6      <1> ; Beginning: 17/01/2016
7      <1> ; -----
8      <1> ; Assembler: NASM version 2.15 (trdos386.s)
9      <1> ; -----
10     <1> ; Turkish Rational DOS
11     <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
12     <1> ; -----
13     <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
14     <1> ; 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> ; Ref: Retro UNIX 386 v1.2 - keyboard.s - 11/06/2022
20     <1> ; -----
21     <1> ; Retro UNIX 386 v1 Kernel - KEYBOARD.INC
22     <1> ; Last Modification: 17/10/2015
23     <1> ; (keyboard Data is in 'KYBDATA.INC')
24     <1> ; -----
25     <1> ; //////////// KEYBOARD FUNCTIONS (PROCEDURES) ////////////
26     <1> ; -----
27     <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
28     <1> ; -----
29     <1> ; 03/12/2014
30     <1> ; 26/08/2014
31     <1> ; KEYBOARD I/O
32     <1> ; (INT_16h - Retro UNIX 8086 v1 - U9.ASM, 30/06/2014)
33     <1> ; -----
34     <1> ; NOTE: 'k0' to 'k7' are name of OPMASK registers.
35     <1> ; (The reason of using '_k' labels!!!) (27/08/2014)
36     <1> ; NOTE: 'NOT' keyword is '~' unary operator in NASM.
37     <1> ; ('NOT LC_HC' --> '~LC_HC') (bit reversing operator)
38     <1> ; -----
39     <1> int16h: ; 30/06/2015
40     <1> ;getc:
41     <1>     pushfd ; 28/08/2014
42     <1>     push cs
43     <1>     call KEYBOARD_IO_1 ; getc_int
44     <1>     retn
45     <1> ; -----
46     <1> ; 24/07/2022 - TRDOS 386 v2.0.5
47     <1> ; -----
48     <1> ; ----- SHIFT STATUS
49     <1> _K3E: ; GET THE EXTENDED SHIFT STATUS FLAGS
50     <1>     mov ah, [KB_FLAG_1] ; GET SYSTEM SHIFT KEY STATUS
51     <1>     and ah, SYS_SHIFT ; MASK ALL BUT SYS KEY BIT
52     <1>     ;mov cl, 5 ; SHIFT THEW SYSTEMKEY BIT OVER TO
53     <1>     ;shl ah, cl ; BIT 7 POSITION
54     <1>     shl ah, 5
55     <1>     mov al, [KB_FLAG_1] ; GET SYSTEM SHIFT STATES BACK
56     <1>     and al, 01110011b ; ELIMINATE SYS SHIFT, HOLD_STATE AND INS_SHIFT
57     <1>     or ah, al ; MERGE REMAINING BITS INTO AH
58     <1>     mov al, [KB_FLAG_3] ; GET RIGHT CTL AND ALT
59     <1>     and al, 00001100b ; ELIMINATE LC_E0 AND LC_E1
60     <1>     or ah, al ; OR THE SHIFT FLAGS TOGETHER
61     <1> _K3:
62     <1>     mov al, [KB_FLAG] ; GET THE SHIFT STATUS FLAGS
63     <1> ; 24/07/2022
64     <1>     jmp short _KIO_EXIT ; RETURN TO CALLER
65     <1> ; -----
66     <1> getc_int:
67     <1> ; 28/02/2015
68     <1> ; 03/12/2014 (derivation from pc-xt-286 bios source code -1986-,
69     <1> ; instead of pc-at bios - 1985-)
70     <1> ; 28/08/2014 (_k1d)
71     <1> ; 30/06/2014
72     <1> ; 03/03/2014
73     <1> ; 28/02/2014
74     <1> ; Derived from "KEYBOARD_IO_1" procedure of IBM "pc-xt-286"
75     <1> ; rombios source code (21/04/1986)
76     <1> ; 'keybd.asm', INT 16H, KEYBOARD_IO
77     <1> ; -----
78     <1> ; KYBD --- 03/06/86 KEYBOARD BIOS
79     <1> ; -----
80     <1> ; --- INT 16 H -----
81     <1> ; KEYBOARD I/O
82     <1> ; THESE ROUTINES PROVIDE READ KEYBOARD SUPPORT
83     <1> ; INPUT
84     <1> ; (AH)= 00H READ THE NEXT ASCII CHARACTER ENTERED FROM THE KEYBOARD,
85     <1> ; RETURN THE RESULT IN (AL), SCAN CODE IN (AH).
86     <1> ; THIS IS THE COMPATIBLE READ INTERFACE, EQUIVALENT TO THE
87     <1> ; STANDARD PC OR PCAT KEYBOARD
88     <1> ; -----
89     <1> ; (AH)= 01H SET THE ZERO FLAG TO INDICATE IF AN ASCII CHARACTER IS
90     <1> ; AVAILABLE TO BE READ FROM THE KEYBOARD BUFFER.
91     <1> ; (ZF)= 1 -- NO CODE AVAILABLE
92     <1> ; (ZF)= 0 -- CODE IS AVAILABLE (AX)= CHARACTER
93     <1> ; IF (ZF)= 0, THE NEXT CHARACTER IN THE BUFFER TO BE READ IS
94     <1> ; IN (AX), AND THE ENTRY REMAINS IN THE BUFFER.
95     <1> ; THIS WILL RETURN ONLY PC/PCAT KEYBOARD COMPATIBLE CODES
96     <1> ; -----
97     <1> ; (AH)= 02H RETURN THE CURRENT SHIFT STATUS IN AL REGISTER
98     <1> ; THE BIT SETTINGS FOR THIS CODE ARE INDICATED IN THE
99     <1> ; EQUATES FOR @KB_FLAG
100    <1> ; -----
101    <1> ; (AH)= 03H SET TYPAMATIC RATE AND DELAY
102    <1> ; (AL) = 05H
103    <1> ; (BL) = TYPAMATIC RATE (BITS 5 - 7 MUST BE RESET TO 0)
104    <1> ; -----
105    <1> ;
106    <1> ; REGISTER RATE REGISTER RATE
107    <1> ; VALUE SELECTED VALUE SELECTED
108    <1> ; -----
109    <1> ; 00H 30.0 10H 7.5
110    <1> ; 01H 26.7 11H 6.7
111    <1> ; 02H 24.0 12H 6.0
112    <1> ; 03H 21.8 13H 5.5
113    <1> ; 04H 20.0 14H 5.0
114    <1> ; 05H 18.5 15H 4.6
115    <1> ; 06H 17.1 16H 4.3
116    <1> ; 07H 16.0 17H 4.0
117    <1> ; 08H 15.0 18H 3.7
118    <1> ; 09H 13.3 19H 3.3
119    <1> ; 0AH 12.0 1AH 3.0
120    <1> ; 0BH 10.9 1BH 2.7
121    <1> ; 0CH 10.0 1CH 2.5
122    <1> ; 0DH 9.2 1DH 2.3
123    <1> ; 0EH 8.6 1EH 2.1
124    <1> ; 0FH 8.0 1FH 2.0
125    <1> ; -----
126    <1> ; (BH) = TYPAMATIC DELAY (BITS 2 - 7 MUST BE RESET TO 0)
127    <1> ; -----
128    <1> ; REGISTER DELAY
129    <1> ; VALUE VALUE

```

```

129 <1> ;-----
130 <1> ; 00H 250 ms
131 <1> ; 01H 500 ms
132 <1> ; 02H 750 ms
133 <1> ; 03H 1000 ms
134 <1> ;-----
135 <1> (AH)= 05H PLACE ASCII CHARACTER/SCAN CODE COMBINATION IN KEYBOARD
136 <1> ; BUFFER AS IF STRUCK FROM KEYBOARD
137 <1> ; ENTRY: (CL) = ASCII CHARACTER
138 <1> ; (CH) = SCAN CODE
139 <1> ; EXIT: (AH) = 00H = SUCCESSFUL OPERATION
140 <1> ; (AL) = 01H = UNSUCCESSFUL - BUFFER FULL
141 <1> ; FLAGS: CARRY IF ERROR
142 <1> ;-----
143 <1> (AH)= 10H EXTENDED READ INTERFACE FOR THE ENHANCED KEYBOARD,
144 <1> ; OTHERWISE SAME AS FUNCTION AH=0
145 <1> ;-----
146 <1> (AH)= 11H EXTENDED ASCII STATUS FOR THE ENHANCED KEYBOARD,
147 <1> ; OTHERWISE SAME AS FUNCTION AH=1
148 <1> ;-----
149 <1> (AH)= 12H RETURN THE EXTENDED SHIFT STATUS IN AX REGISTER
150 <1> ; AL = BITS FROM KB_FLAG, AH = BITS FOR LEFT AND RIGHT
151 <1> ; CTL AND ALT KEYS FROM KB_FLAG_1 AND KB_FLAG_3
152 <1> ;-----
153 <1> ; OUTPUT
154 <1> ; AS NOTED ABOVE, ONLY (AX) AND FLAGS CHANGED
155 <1> ; ALL REGISTERS RETAINED
156 <1> ;-----
157 <1> ; 07/08/2022
158 <1> ; 24/07/2022 - TRDOS 386 v2.0.5
159 <1> ; 12/04/2021 - TRDOS 386 v2.0.3 (32 bit push/pop)
160 <1> ; 15/01/2017
161 <1> ; 14/01/2017
162 <1> ; 02/01/2017
163 <1> ; 29/05/2016
164 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
165 <1> int32h: ; Keyboard BIOS
166 <1>
167 <1> KEYBOARD_IO_1:
168 <1> ; sti ; INTERRUPTS BACK ON
169 <1> ; 29/05/2016
170 <1> and byte [esp+8], 1011110b ; clear zero flag and cary flag
171 <1> ;
172 <1> push ds ; SAVE CURRENT DS
173 <1> push ebx ; SAVE BX TEMPORARILY
174 <1> ;push ecx ; SAVE CX TEMPORARILY
175 <1> mov bx, KDATA
176 <1> mov ds, bx ; PUT SEGMENT VALUE OF DATA AREA INTO DS
177 <1> ; 14/01/2017
178 <1> mov ebx, [esp]
179 <1> ; 15/01/2017
180 <1> ; 02/01/2017
181 <1> ;mov byte [intflg], 32h ; keyboard interrupt
182 <1> sti
183 <1> ;
184 <1> or ah, ah ; CHECK FOR (AH)= 00H
185 <1> jz short _K1 ; ASCII_READ
186 <1> dec ah ; CHECK FOR (AH)= 01H
187 <1> jz short _K2 ; ASCII_STATUS
188 <1> dec ah ; CHECK FOR (AH)= 02H
189 <1> jz short _K3 ; SHIFT STATUS
190 <1> dec ah ; CHECK FOR (AH)= 03H
191 <1> jz short _K300 ; SET TYPAMATIC RATE/DELAY
192 <1> sub ah, 2 ; CHECK FOR (AH)= 05H
193 <1> ;jz short _K500 ; KEYBOARD WRITE
194 <1> ; 07/08/2022
195 <1> jnz short _KIO1
196 <1> jmp _K500
197 <1> _KIO1:
198 <1> sub ah, 11 ; AH = 10H
199 <1> jz short _K1E ; EXTENDED ASCII READ
200 <1> dec ah ; CHECK FOR (AH)= 11H
201 <1> jz short _K2E ; EXTENDED_ASCII_STATUS
202 <1> dec ah ; CHECK FOR (AH)= 12H
203 <1> jz short _K3E ; EXTENDED_SHIFT_STATUS
204 <1> _KIO_EXIT:
205 <1> ; 02/01/2017
206 <1> cli
207 <1> ;mov byte [intflg], 0 ; 15/01/2017
208 <1> ;
209 <1> ;pop ecx ; RECOVER REGISTER
210 <1> pop ebx ; RECOVER REGISTER
211 <1> pop ds ; RECOVER SEGMENT
212 <1> iretd ; INVALID COMMAND, EXIT
213 <1>
214 <1> ; 24/07/2022
215 <1> ;
216 <1> ;----- SHIFT STATUS
217 <1> ;_K3E: ; GET THE EXTENDED SHIFT STATUS FLAGS
218 <1> ; mov ah, [KB_FLAG_1] ; GET SYSTEM SHIFT KEY STATUS
219 <1> ; and ah, SYS_SHIFT ; MASK ALL BUT SYS KEY BIT
220 <1> ; ;mov cl, 5 ; SHIFT THEW SYSTEMKEY BIT OVER TO
221 <1> ; ;shl ah, cl ; BIT 7 POSITION
222 <1> ; shl ah, 5
223 <1> ; mov al, [KB_FLAG_1] ; GET SYSTEM SHIFT STATES BACK
224 <1> ; and al, 01110011b ; ELIMINATE SYS SHIFT, HOLD_STATE AND INS_SHIFT
225 <1> ; or ah, al ; MERGE REMAINING BITS INTO AH
226 <1> ; mov al, [KB_FLAG_3] ; GET RIGHT CTL AND ALT
227 <1> ; and al, 00001100b ; ELIMINATE LC_E0 AND LC_E1
228 <1> ; or ah, al ; OR THE SHIFT FLAGS TOGETHER
229 <1> ;_K3:
230 <1> ; mov al, [KB_FLAG] ; GET THE SHIFT STATUS FLAGS
231 <1> ; ; 24/07/2022
232 <1> ; jmp short _KIO_EXIT ; RETURN TO CALLER
233 <1>
234 <1> ;----- ASCII CHARACTER
235 <1> ;_K1E:
236 <1> call _K1S ; GET A CHARACTER FROM THE BUFFER (EXTENDED)
237 <1> call _KIO_E_XLAT ; ROUTINE TO XLATE FOR EXTENDED CALLS
238 <1> jmp short _KIO_EXIT ; GIVE IT TO THE CALLER
239 <1>
240 <1> _K1: call _K1S ; GET A CHARACTER FROM THE BUFFER
241 <1> call _KIO_S_XLAT ; ROUTINE TO XLATE FOR STANDARD CALLS
242 <1> jc short _K1 ; CARRY SET MEANS TROW CODE AWAY
243 <1> _K1A:
244 <1> jmp short _KIO_EXIT ; RETURN TO CALLER
245 <1>
246 <1> ;----- ASCII STATUS
247 <1> ;_K2E:
248 <1> call _K2S ; TEST FOR CHARACTER IN BUFFER (EXTENDED)
249 <1> jz short _K2B ; RETURN IF BUFFER EMPTY
250 <1> pushf ; SAVE ZF FROM TEST
251 <1> call _KIO_E_XLAT ; ROUTINE TO XLATE FOR EXTENDED CALLS
252 <1> jmp short _K2A ; GIVE IT TO THE CALLER

```

```

253      <1> _K2:
254      <1> call    _K2S                ; TEST FOR CHARACTER IN BUFFER
255      <1> jz     short _K2B          ; RETURN IF BUFFER EMPTY
256      <1> pushf                    ; SAVE ZF FROM TEST
257      <1> call    _KIO_S_XLAT        ; ROUTINE TO XLATE FOR STANDARD CALLS
258      <1> jnc     short _K2A          ; CARRY CLEAR MEANS PASS VALID CODE
259      <1> popf                      ; INVALID CODE FOR THIS TYPE OF CALL
260      <1> call    _K1S                ; THROW THE CHARACTER AWAY
261      <1> jmp     short _K2           ; GO LOOK FOR NEXT CHAR, IF ANY
262      <1> _K2A:
263      <1> popf                      ; RESTORE ZF FROM TEST
264      <1> _K2B:
265      <1> ; 02/01/2017
266      <1> cli
267      <1> ;; mov byte [intflg], 0 ;; 15/01/2017
268      <1> ;
269      <1> ;pop    ecx                ; RECOVER REGISTER
270      <1> pop     ebx                ; RECOVER REGISTER
271      <1> pop     ds                 ; RECOVER SEGMENT
272      <1> ; (*) 29/05/2016
273      <1> ; (*) retf 4                ; THROW AWAY (e)FLAGS
274      <1> jc     short _k2d
275      <1> jnz     short _k2c
276      <1> or      byte [esp+8], 0100000b ; set zero flag bit of eflags register
277      <1> _k2c:
278      <1> iretd
279      <1> _k2d:
280      <1> ; 29/05/2016 -set carry flag on stack-
281      <1> ; [esp] = EIP
282      <1> ; [esp+4] = CS
283      <1> ; [esp+8] = E-FLAGS
284      <1> or      byte [esp+8], 1 ; set carry bit of eflags register
285      <1> ; [esp+12] = ESP (user)
286      <1> ; [esp+16] = SS (User)
287      <1> iretd
288      <1> ;
289      <1> ; (*) 29/05/2016 - 'retf 4' intruction causes to stack fault
290      <1> ; (OUTER-PRIVILEGE-LEVEL)
291      <1> ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986
292      <1> ; // RETF instruction:
293      <1> ;
294      <1> ; IF OperandMode=32 THEN
295      <1> ; Load CS:EIP from stack;
296      <1> ; Set CS RPL to CPL;
297      <1> ; Increment ESP by 8 plus the immediate offset if it exists;
298      <1> ; Load SS:ESP from stack;
299      <1> ; ELSE (* OperandMode=16 *)
300      <1> ; Load CS:IP from stack;
301      <1> ; Set CS RPL to CPL;
302      <1> ; Increment SP by 4 plus the immediate offset if it exists;
303      <1> ; Load SS:SP from stack;
304      <1> ; FI;
305      <1> ;
306      <1> ; //
307      <1> ;
308      <1> ; 24/07/2022
309      <1> ;----- SET TYPAMATIC RATE AND DELAY
310      <1> _K300:
311      <1> cmp     al, 5                ; CORRECT FUNCTION CALL?
312      <1> jne     short _KIO_EXIT      ; NO, RETURN
313      <1> test    bl, 0E0h            ; TEST FOR OUT-OF-RANGE RATE
314      <1> jnz     short _KIO_EXIT      ; RETURN IF SO
315      <1> test    bh, 0FCh            ; TEST FOR OUT-OF-RANGE DELAY
316      <1> jnz     short _KIO_EXIT      ; RETURN IF SO
317      <1> mov     al, KB_TYPA_RD       ; COMMAND FOR TYPAMATIC RATE/DELAY
318      <1> call    SND_DATA             ; SEND TO KEYBOARD
319      <1> ;mov     cx, 5                ; SHIFT COUNT
320      <1> ;shl     bh, cl               ; SHIFT DELAY OVER
321      <1> shl     bh, 5
322      <1> mov     al, bl               ; PUT IN RATE
323      <1> or      al, bh               ; AND DELAY
324      <1> call    SND_DATA             ; SEND TO KEYBOARD
325      <1> jmp     _KIO_EXIT            ; RETURN TO CALLER
326      <1> ;
327      <1> ;----- WRITE TO KEYBOARD BUFFER
328      <1> _K500:
329      <1> push    esi                 ; SAVE SI (esi)
330      <1> cli
331      <1> mov     ebx, [BUFFER_TAIL]    ; GET THE 'IN TO' POINTER TO THE BUFFER
332      <1> mov     esi, ebx
333      <1> call    _K4                  ; SAVE A COPY IN CASE BUFFER NOT FULL
334      <1> cmp     ebx, [BUFFER_HEAD]   ; BUMP THE POINTER TO SEE IF BUFFER IS FULL
335      <1> jbe     short _K502           ; WILL THE BUFFER OVERRUN IF WE STORE THIS?
336      <1> mov     [esi], cx            ; YES - INFORM CALLER OF ERROR
337      <1> mov     [BUFFER_TAIL], ebx   ; NO - PUT ASCII/SCAN CODE INTO BUFFER
338      <1> sub     al, al              ; ADJUST 'IN TO' POINTER TO REFLECT CHANGE
339      <1> jmp     short _K504           ; TELL CALLER THAT OPERATION WAS SUCCESSFUL
340      <1> _K502:
341      <1> mov     al, 01h              ; SUB INSTRUCTION ALSO RESETS CARRY FLAG
342      <1> _K504:
343      <1> sti
344      <1> pop     esi                 ; RECOVER SI (esi)
345      <1> jmp     _KIO_EXIT            ; RETURN TO CALLER WITH STATUS IN AL
346      <1> ;
347      <1> ;----- READ THE KEY TO FIGURE OUT WHAT TO DO -----
348      <1> _K1S:
349      <1> cli ; 03/12/2014
350      <1> mov     ebx, [BUFFER_HEAD]   ; GET POINTER TO HEAD OF BUFFER
351      <1> cmp     ebx, [BUFFER_TAIL]   ; TEST END OF BUFFER
352      <1> jne     short _K1U            ; IF ANYTHING IN BUFFER SKIP INTERRUPT
353      <1> jne     short _k1x ; 03/12/2014
354      <1> ;
355      <1> ; 03/12/2014
356      <1> ; 28/08/2014
357      <1> ; PERFORM OTHER FUNCTION ?? here !
358      <1> ; MOV AX, 9002h              ; MOVE IN WAIT CODE & TYPE
359      <1> ; INT 15h                    ; PERFORM OTHER FUNCTION
360      <1> _K1T:
361      <1> sti ; ASCII READ
362      <1> nop ; INTERRUPTS BACK ON DURING LOOP
363      <1> _K1U:
364      <1> cli ; INTERRUPTS BACK OFF
365      <1> mov     ebx, [BUFFER_HEAD]   ; GET POINTER TO HEAD OF BUFFER
366      <1> cmp     ebx, [BUFFER_TAIL]   ; TEST END OF BUFFER
367      <1> _k1x:
368      <1> push    ebx                 ; SAVE ADDRESS
369      <1> pushf                    ; SAVE FLAGS
370      <1> call    MAKE_LED             ; GO GET MODE INDICATOR DATA BYTE
371      <1> mov     bl, [KB_FLAG_2]      ; GET PREVIOUS BITS
372      <1> xor     bl, al               ; SEE IF ANY DIFFERENT
373      <1> and     bl, 07h ; KB_LEDS   ; ISOLATE INDICATOR BITS
374      <1> jz     short _K1V            ; IF NO CHANGE BYPASS UPDATE
375      <1> call    SND_LED1
376      <1> cli ; DISABLE INTERRUPTS

```



```

377 <1> _K1v:
378 00001061 9D <1> popf ; RESTORE FLAGS
379 00001062 5B <1> pop ebx ; RESTORE ADDRESS
380 00001063 74D3 <1> je short _K1T ; LOOP UNTIL SOMETHING IN BUFFER
381 <1> ;
382 00001065 668B03 <1> mov ax, [ebx] ; GET SCAN CODE AND ASCII CODE
383 00001068 E86F000000 <1> call _K4 ; MOVE POINTER TO NEXT POSITION
384 0000106D 891D[F6670000] <1> mov [BUFFER_HEAD], ebx ; STORE VALUE IN VARIABLE
385 00001073 C3 <1> retn ; RETURN
386 <1>
387 <1> ;----- READ THE KEY TO SEE IF ONE IS PRESENT -----
388 <1> _K2S:
389 00001074 FA <1> cli ; INTERRUPTS OFF
390 00001075 881D[F6670000] <1> mov ebx, [BUFFER_HEAD] ; GET HEAD POINTER
391 0000107B 381D[FA670000] <1> cmp ebx, [BUFFER_TAIL] ; IF EQUAL (Z=1) THEN NOTHING THERE
392 00001081 668B03 <1> mov ax, [ebx]
393 00001084 9C <1> pushf ; SAVE FLAGS
394 <1> ;push ax ; SAVE CODE
395 <1> ; 12/04/2021
396 00001085 50 <1> push eax
397 00001086 E88C060000 <1> call MAKE_LED ; GO GET MODE INDICATOR DATA BYTE
398 00001088 8A1D[EB670000] <1> mov bl, [KB_FLAG_2] ; GET PREVIOUS BITS
399 00001091 30C3 <1> xor bl, al ; SEE IF ANY DIFFERENT
400 00001093 80E307 <1> and bl, 07h ; KB_LEDS ; ISOLATE INDICATOR BITS
401 00001096 7405 <1> jz short _K2T ; IF NO CHANGE BYPASS UPDATE
402 00001098 E80F060000 <1> call SND_LED ; GO TURN ON MODE INDICATORS
403 <1> _K2T:
404 <1> ;pop ax ; RESTORE CODE
405 <1> ; 12/04/2021
406 0000109D 58 <1> pop eax
407 0000109E 9D <1> popf ; RESTORE FLAGS
408 0000109F FB <1> sti ; INTERRUPTS BACK ON
409 000010A0 C3 <1> retn ; RETURN
410 <1>
411 <1> ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR EXTENDED CALLS -----
412 <1> _KIO_E_XLAT:
413 000010A1 3CF0 <1> cmp al, 0F0h ; IS IT ONE OF THE FILL-INS?
414 000010A3 7506 <1> jne short _KIO_E_RET ; NO, PASS IT ON
415 000010A5 08E4 <1> or ah, ah ; AH = 0 IS SPECIAL CASE
416 000010A7 7402 <1> jz short _KIO_E_RET ; PASS THIS ON UNCHANGED
417 000010A9 30C0 <1> xor al, al ; OTHERWISE SET AL = 0
418 <1> _KIO_E_RET:
419 000010AB C3 <1> retn ; GO BACK
420 <1>
421 <1> ;----- ROUTINE TO TRANSLATE SCAN CODE PAIRS FOR STANDARD CALLS -----
422 <1> _KIO_S_XLAT:
423 000010AC 80FCE0 <1> cmp ah, 0E0h ; IS IT KEYPAD ENTER OR / ?
424 000010AF 750F <1> jne short _KIO_S2 ; NO, CONTINUE
425 000010B1 3C0D <1> cmp al, 0Dh ; KEYPAD ENTER CODE?
426 000010B3 7408 <1> je short _KIO_S1 ; YES, MESSAGE A BIT
427 000010B5 3C0A <1> cmp al, 0Ah ; CTRL KEYPAD ENTER CODE?
428 000010B7 7404 <1> je short _KIO_S1 ; YES, MESSAGE THE SAME
429 000010B9 B435 <1> mov ah, 35h ; NO, MUST BE KEYPAD /
430 <1> _kio_ret: ; 03/12/2014
431 000010BB F8 <1> clc
432 000010BC C3 <1> retn
433 <1> ;jmp short _KIO_USE ; GIVE TO CALLER
434 <1> _KIO_S1:
435 000010BD B41C <1> mov ah, 1Ch ; CONVERT TO COMPATIBLE OUTPUT
436 <1> ;jmp short _KIO_USE ; GIVE TO CALLER
437 000010BF C3 <1> retn
438 <1> _KIO_S2:
439 000010C0 80FC84 <1> cmp ah, 84h ; IS IT ONE OF EXTENDED ONES?
440 000010C3 7715 <1> ja short _KIO_DIS ; YES, THROW AWAY AND GET ANOTHER CHAR
441 000010C5 3CF0 <1> cmp al, 0F0h ; IS IT ONE OF THE FILL-INS?
442 000010C7 7506 <1> jne short _KIO_S3 ; NO, TRY LAST TEST
443 000010C9 08E4 <1> or ah, ah ; AH = 0 IS SPECIAL CASE
444 000010CB 740C <1> jz short _KIO_USE ; PASS THIS ON UNCHANGED
445 000010CD EB0B <1> jmp short _KIO_DIS ; THROW AWAY THE REST
446 <1> _KIO_S3:
447 000010CF 3CE0 <1> cmp al, 0E0h ; IS IT AN EXTENSION OF A PREVIOUS ONE?
448 <1> ;jne short _KIO_USE ; NO, MUST BE A STANDARD CODE
449 000010D1 75E8 <1> jne short _kio_ret
450 000010D3 08E4 <1> or ah, ah ; AH = 0 IS SPECIAL CASE
451 000010D5 7402 <1> jz short _KIO_USE ; JUMP IF AH = 0
452 000010D7 30C0 <1> xor al, al ; CONVERT TO COMPATIBLE OUTPUT
453 <1> ;jmp short _KIO_USE ; PASS IT ON TO CALLER
454 <1> _KIO_USE:
455 <1> ;clc ; CLEAR CARRY TO INDICATE GOOD CODE
456 000010D9 C3 <1> retn ; RETURN
457 <1> _KIO_DIS:
458 000010DA F9 <1> stc ; SET CARRY TO INDICATE DISCARD CODE
459 000010DB C3 <1> retn ; RETURN
460 <1>
461 <1> ;----- INCREMENT BUFFER POINTER ROUTINE -----
462 <1> _K4:
463 000010DC 43 <1> inc ebx
464 000010DD 43 <1> inc ebx ; MOVE TO NEXT WORD IN LIST
465 000010DE 3B1D[EE670000] <1> cmp ebx, [BUFFER_END] ; AT END OF BUFFER?
466 <1> ;jne short _K5 ; NO, CONTINUE
467 000010E4 7206 <1> jb short _K5
468 000010E6 8B1D[EE670000] <1> mov ebx, [BUFFER_START] ; YES, RESET TO BUFFER BEGINNING
469 <1> _K5:
470 000010EC C3 <1> retn
471 <1>
472 <1> ; 20/02/2015
473 <1> ; 05/12/2014
474 <1> ; 26/08/2014
475 <1> ; KEYBOARD (HARDWARE) INTERRUPT - IRQ LEVEL 1
476 <1> ; (INT_09h - Retro UNIX 8086 v1 - U9.ASM, 07/03/2014)
477 <1> ;
478 <1> ; Derived from "KB_INT_1" procedure of IBM "pc-at"
479 <1> ; rombios source code (06/10/1985)
480 <1> ; 'keybd.asm', HARDWARE INT 09h - (IRQ Level 1)
481 <1>
482 <1> ; EQUATES (IBM PC-XT-286 BIOS, 1986, 'POSSEQU.INC')
483 <1>
484 <1> ;----- 8042 COMMANDS -----
485 <1> ENA_KBD equ 0AEh ; ENABLE KEYBOARD COMMAND
486 <1> DIS_KBD equ 0ADh ; DISABLE KEYBOARD COMMAND
487 <1> SHUT_CMD equ 0FEh ; CAUSE A SHUTDOWN COMMAND
488 <1> ;----- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----
489 <1> STATUS_PORT equ 064h ; 8042 STATUS PORT
490 <1> INPT_BUF_FULL equ 00000010b ; 1 = +INPUT BUFFER FULL
491 <1> PORT_A equ 060h ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
492 <1> ;----- 8042 KEYBOARD RESPONSE -----
493 <1> KB_ACK equ 0FAh ; ACKNOWLEDGE PROM TRANSMISSION
494 <1> KB_RESEND equ 0FEh ; RESEND REQUEST
495 <1> KB_OVER_RUN equ 0FFh ; OVER RUN SCAN CODE
496 <1> ;----- KEYBOARD/LED COMMANDS -----
497 <1> KB_ENABLE equ 0F4h ; KEYBOARD ENABLE
498 <1> LED_CMD equ 0EDh ; LED WRITE COMMAND
499 <1> KB_TYPA_RD equ 0F3h ; TYPAMATIC RATE/DELAY COMMAND
500 <1> ;----- KEYBOARD SCAN CODES -----

```

```

501 <1> NUM_KEY equ 69 ; SCAN CODE FOR NUMBER LOCK KEY
502 <1> SCROLL_KEY equ 70 ; SCAN CODE FOR SCROLL LOCK KEY
503 <1> ALT_KEY equ 56 ; SCAN CODE FOR ALTERNATE SHIFT KEY
504 <1> CTL_KEY equ 29 ; SCAN CODE FOR CONTROL KEY
505 <1> CAPS_KEY equ 58 ; SCAN CODE FOR SHIFT LOCK KEY
506 <1> DEL_KEY equ 83 ; SCAN CODE FOR DELETE KEY
507 <1> INS_KEY equ 82 ; SCAN CODE FOR INSERT KEY
508 <1> LEFT_KEY equ 42 ; SCAN CODE FOR LEFT SHIFT
509 <1> RIGHT_KEY equ 54 ; SCAN CODE FOR RIGHT SHIFT
510 <1> SYS_KEY equ 84 ; SCAN CODE FOR SYSTEM KEY
511 <1> ;----- ENHANCED KEYBOARD SCAN CODES -----
512 <1> ID_1 equ 0ABh ; 1ST ID CHARACTER FOR KBX
513 <1> ID_2 equ 041h ; 2ND ID CHARACTER FOR KBX
514 <1> ID_2A equ 054h ; ALTERNATE 2ND ID CHARACTER FOR KBX
515 <1> F11_M equ 87 ; F11 KEY MAKE
516 <1> F12_M equ 88 ; F12 KEY MAKE
517 <1> MC_E0 equ 224 ; GENERAL MARKER CODE
518 <1> MC_E1 equ 225 ; PAUSE KEY MARKER CODE
519 <1> ;----- FLAG EQUATES WITHIN @KB_FLAG -----
520 <1> RIGHT_SHIFT equ 0000001b ; RIGHT SHIFT KEY DEPRESSED
521 <1> LEFT_SHIFT equ 00000010b ; LEFT SHIFT KEY DEPRESSED
522 <1> CTL_SHIFT equ 00000100b ; CONTROL SHIFT KEY DEPRESSED
523 <1> ALT_SHIFT equ 00001000b ; ALTERNATE SHIFT KEY DEPRESSED
524 <1> SCROLL_STATE equ 00010000b ; SCROLL LOCK STATE IS ACTIVE
525 <1> NUM_STATE equ 00100000b ; NUM LOCK STATE IS ACTIVE
526 <1> CAPS_STATE equ 01000000b ; CAPS LOCK STATE IS ACTIVE
527 <1> INS_STATE equ 10000000b ; INSERT STATE IS ACTIVE
528 <1> ;----- FLAG EQUATES WITHIN @KB_FLAG_1 -----
529 <1> L_CTL_SHIFT equ 00000001b ; LEFT CTL KEY DOWN
530 <1> L_ALT_SHIFT equ 00000010b ; LEFT ALT KEY DOWN
531 <1> SYS_SHIFT equ 00000100b ; SYSTEM KEY DEPRESSED AND HELD
532 <1> HOLD_STATE equ 00001000b ; SUSPEND KEY HAS BEEN TOGGLED
533 <1> SCROLL_SHIFT equ 00010000b ; SCROLL LOCK KEY IS DEPRESSED
534 <1> NUM_SHIFT equ 00100000b ; NUM LOCK KEY IS DEPRESSED
535 <1> CAPS_SHIFT equ 01000000b ; CAPS LOCK KEY IS DEPRESSED
536 <1> INS_SHIFT equ 10000000b ; INSERT KEY IS DEPRESSED
537 <1> ;----- FLAGS EQUATES WITHIN @KB_FLAG_2 -----
538 <1> KB_LEDS equ 00000111b ; KEYBOARD LED STATE BITS
539 <1> ;
540 <1> ; equ 00000001b ; SCROLL LOCK INDICATOR
541 <1> ; equ 00000010b ; NUM LOCK INDICATOR
542 <1> ; equ 00000100b ; CAPS LOCK INDICATOR
543 <1> ; equ 00001000b ; RESERVED (MUST BE ZERO)
544 <1> KB_FA equ 00010000b ; ACKNOWLEDGMENT RECEIVED
545 <1> KB_FE equ 00100000b ; RESEND RECEIVED FLAG
546 <1> KB_PR_LED equ 01000000b ; MODE INDICATOR UPDATE
547 <1> KB_ERR equ 10000000b ; KEYBOARD TRANSMIT ERROR FLAG
548 <1> ;----- FLAGS EQUATES WITHIN @KB_FLAG_3 -----
549 <1> LC_E1 equ 00000001b ; LAST CODE WAS THE E1 HIDDEN CODE
550 <1> LC_E0 equ 00000010b ; LAST CODE WAS THE E0 HIDDEN CODE
551 <1> R_CTL_SHIFT equ 00000100b ; RIGHT CTL KEY DOWN
552 <1> R_ALT_SHIFT equ 00001000b ; RIGHT ALT KEY DOWN
553 <1> GRAPH_ON equ 00001000b ; ALT GRAPHICS KEY DOWN (WT ONLY)
554 <1> KBX equ 00010000b ; ENHANCED KEYBOARD INSTALLED
555 <1> SET_NUM_LK equ 00100000b ; FORCE NUM LOCK IF READ ID AND KBX
556 <1> LC_AB equ 01000000b ; LAST CHARACTER WAS FIRST ID CHARACTER
557 <1> RD_ID equ 10000000b ; DOING A READ ID (MUST BE BIT0)
558 <1> ;
559 <1> ;----- INTERRUPT EQUATES -----
560 <1> EOI equ 020h ; END OF INTERRUPT COMMAND TO 8259
561 <1> INTA00 equ 020h ; 8259 PORT
562 <1>
563 <1> kb_int:
564 <1>
565 <1> ; 24/07/2022 - TRDOS 386 v2.0.5
566 <1> ; 12/04/2021 - TRDOS 386 v2.0.3 (32 bit push/pop)
567 <1> ; 17/10/2015 ('ctrlbrk')
568 <1> ; 05/12/2014
569 <1> ; 04/12/2014 (derived from pc-xt-286 bios source code -1986-)
570 <1> ; 26/08/2014
571 <1>
572 <1> ; 03/06/86 KEYBOARD BIOS
573 <1>
574 <1> ;--- HARDWARE INT 09H -- (IRQ LEVEL 1) -----
575 <1>
576 <1> ; KEYBOARD INTERRUPT ROUTINE
577 <1> ;
578 <1> ;-----
579 <1>
580 <1> KB_INT_1:
581 <1> sti ; ENABLE INTERRUPTS
582 <1> ;push ebp
583 <1> push eax
584 <1> push ebx
585 <1> push ecx
586 <1> push edx
587 <1> push esi
588 <1> push edi
589 <1> push ds
590 <1> push es
591 <1> cld ; FORWARD DIRECTION
592 <1> mov ax, KDATA
593 <1> mov ds, ax
594 <1> mov es, ax
595 <1> ;
596 <1> ;----- WAIT FOR KEYBOARD DISABLE COMMAND TO BE ACCEPTED
597 <1> mov al, DIS_KBD ; DISABLE THE KEYBOARD COMMAND
598 <1> call SHIP_IT ; EXECUTE DISABLE
599 <1> cli ; DISABLE INTERRUPTS
600 <1> mov ecx, 10000h ; SET MAXIMUM TIMEOUT
601 <1> KB_INT_01:
602 <1> in al, STATUS_PORT ; READ ADAPTER STATUS
603 <1> test al, INPT_BUF_FULL ; CHECK INPUT BUFFER FULL STATUS BIT
604 <1> loopnz KB_INT_01 ; WAIT FOR COMMAND TO BE ACCEPTED
605 <1> ;
606 <1> ;----- READ CHARACTER FROM KEYBOARD INTERFACE
607 <1> in al, PORT_A ; READ IN THE CHARACTER
608 <1> ;
609 <1> ;----- SYSTEM HOOK INT 15H - FUNCTION 4FH (ON HARDWARE INT LEVEL 9H)
610 <1> mov ah, 04Fh ; SYSTEM INTERCEPT - KEY CODE FUNCTION
611 <1> stc ; SET CY=1 (IN CASE OF IRET)
612 <1> int 15h ; CASSETTE CALL (AL)=KEY SCAN CODE
613 <1> ; RETURNS CY=1 FOR INVALID FUNCTION
614 <1> jc KB_INT_02 ; CONTINUE IF CARRY FLAG SET ((AL)=CODE)
615 <1> jmp k26 ; EXIT IF SYSTEM HANDLES SCAN CODE
616 <1> ; EXIT HANDLES HARDWARE EOI AND ENABLE
617 <1> ;
618 <1> ;----- CHECK FOR A RESEND COMMAND TO KEYBOARD
619 <1> KB_INT_02: ; (AL)= SCAN CODE
620 <1> sti ; ENABLE INTERRUPTS AGAIN
621 <1> cmp al, KB_RESEND ; IS THE INPUT A RESEND
622 <1> je short KB_INT_4 ; GO IF RESEND
623 <1> ;
624 <1> ;----- CHECK FOR RESPONSE TO A COMMAND TO KEYBOARD

```

```

625 00001119 3CFA      <1>      cmp     al, KB_ACK      ; IS THE INPUT AN ACKNOWLEDGE
626 0000111B 7514      <1>      jne     short KB_INT_2      ; GO IF NOT
627      <1>      ;
628      <1>      ;----- A COMMAND TO THE KEYBOARD WAS ISSUED
629 0000111D FA        <1>      cli     ; DISABLE INTERRUPTS
630 0000111E 800D[EB670000]10 <1>      or     byte [KB_FLAG_2], KB_FA ; INDICATE ACK RECEIVED
631      <1>      ; jmp     K26      ; RETURN IF NOT ACK RETURNED FOR DATA)
632      <1>      ; 12/04/2021
633 00001125 EB76      <1>      jmp     short ID_EX ; K26
634      <1>      ;
635      <1>      ;----- RESEND THE LAST BYTE
636      <1>      KB_INT_4:
637 00001127 FA        <1>      cli     ; DISABLE INTERRUPTS
638 00001128 800D[EB670000]20 <1>      or     byte [KB_FLAG_2], KB_FE ; INDICATE RESEND RECEIVED
639      <1>      ; jmp     K26      ; RETURN IF NOT ACK RETURNED FOR DATA)
640      <1>      ; 12/04/2021
641 0000112F EB6C      <1>      jmp     short ID_EX ; K26
642      <1>      ;
643      <1>      ;----- UPDATE MODE INDICATORS IF CHANGE IN STATE
644      <1>      KB_INT_2:
645      <1>      ; push    ax      ; SAVE DATA IN
646      <1>      ; 12/04/2021
647 00001131 50        <1>      push    eax
648 00001132 E8E0050000 <1>      call   MAKE_LED      ; GO GET MODE INDICATOR DATA BYTE
649 00001137 8A1D[EB670000] <1>      mov     bl, [KB_FLAG_2] ; GET PREVIOUS BITS
650 0000113D 30C3      <1>      xor     bl, al      ; SEE IF ANY DIFFERENT
651 0000113F 80E307     <1>      and     bl, KB_LEDS ; ISOLATE INDICATOR BITS
652 00001142 7405      <1>      jz      short UP0 ; IF NO CHANGE BYPASS UPDATE
653 00001144 E863050000 <1>      call   SND_LED      ; GO TURN ON MODE INDICATORS
654      <1>      UP0:
655      <1>      ; pop     ax      ; RESTORE DATA IN
656      <1>      ; 12/04/2021
657 00001149 58        <1>      pop     eax
658      <1>      ;-----
659      <1>      ; START OF KEY PROCESSING
660      <1>      ;-----
661 0000114A 88C4      <1>      mov     ah, al      ; SAVE SCAN CODE IN AH ALSO
662      <1>      ;
663      <1>      ;----- TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
664 0000114C 3CFF      <1>      cmp     al, KB_OVER_RUN ; IS THIS AN OVERRUN CHAR
665      <1>      ; je     K62      ; BUFFER_FULL_BEEP
666      <1>      ; 12/04/2021
667 0000114E 7505      <1>      jne     short K16
668 00001150 E9E8040000 <1>      jmp     K62
669      <1>      K16:
670 00001155 8A3D[EC670000] <1>      mov     bh, [KB_FLAG_3] ; LOAD FLAGS FOR TESTING
671      <1>      ;
672      <1>      ;----- TEST TO SEE IF A READ_ID IS IN PROGRESS
673 0000115B F6C7C0     <1>      test    bh, RD_ID+LC_AB ; ARE WE DOING A READ ID?
674 0000115E 7442      <1>      jz      short NOT_ID ; CONTINUE IF NOT
675 00001160 7914      <1>      jns     short TST_ID_2 ; IS THE RD_ID FLAG ON?
676 00001162 3CAB      <1>      cmp     al, ID_1 ; IS THIS THE 1ST ID CHARACTER?
677 00001164 7507      <1>      jne     short RST_RD_ID
678 00001166 800D[EC670000]40 <1>      or     byte [KB_FLAG_3], LC_AB ; INDICATE 1ST ID WAS OK
679      <1>      RST_RD_ID:
680 0000116D 8025[EC670000]7F <1>      and     byte [KB_FLAG_3], ~RD_ID ; RESET THE READ ID FLAG
681 00001174 EB27      <1>      jmp     short ID_EX ; AND EXIT
682      <1>      ; 12/04/2021
683      <1>      ; jmp     K26
684      <1>      ;
685      <1>      TST_ID_2:
686 00001176 8025[EC670000]BF <1>      and     byte [KB_FLAG_3], ~LC_AB ; RESET FLAG
687 0000117D 3C54      <1>      cmp     al, ID_2A ; IS THIS THE 2ND ID CHARACTER?
688 0000117F 7415      <1>      je     short KX_BIT ; JUMP IF SO
689 00001181 3C41      <1>      cmp     al, ID_2 ; IS THIS THE 2ND ID CHARACTER?
690 00001183 7518      <1>      jne     short ID_EX ; LEAVE IF NOT
691      <1>      ; 12/04/2021
692      <1>      ; jne     K26
693      <1>      ;
694      <1>      ;----- A READ ID SAID THAT IT WAS ENHANCED KEYBOARD
695 00001185 F6C720     <1>      test    bh, SET_NUM_LK ; SHOULD WE SET NUM LOCK?
696 00001188 740C      <1>      jz      short KX_BIT ; EXIT IF NOT
697 0000118A 800D[E9670000]20 <1>      or     byte [KB_FLAG], NUM_STATE ; FORCE NUM LOCK ON
698 00001191 E816050000 <1>      call   SND_LED      ; GO SET THE NUM LOCK INDICATOR
699      <1>      KX_BIT:
700 00001196 800D[EC670000]10 <1>      or     byte [KB_FLAG_3], KBX ; INDICATE ENHANCED KEYBOARD WAS FOUND
701 0000119D E9CB010000 <1>      jmp     K26 ; EXIT
702      <1>      ;
703      <1>      NOT_ID:
704 000011A2 3CE0      <1>      cmp     al, MC_E0 ; IS THIS THE GENERAL MARKER CODE?
705 000011A4 7509      <1>      jne     short TEST_E1
706 000011A6 800D[EC670000]12 <1>      or     byte [KB_FLAG_3], LC_E0+KBX ; SET FLAG BIT, SET KBX, AND
707 000011AD EB0B      <1>      jmp     short EXIT ; THROW AWAY THIS CODE
708      <1>      ; 12/04/2021
709      <1>      ; jmp     K26A
710      <1>      TEST_E1:
711 000011AF 3CE1      <1>      cmp     al, MC_E1 ; IS THIS THE PAUSE KEY?
712 000011B1 750C      <1>      jne     short NOT_HC
713 000011B3 800D[EC670000]11 <1>      or     byte [KB_FLAG_3], LC_E1+KBX ; SET FLAG BIT, SET KBX, AND
714 000011BA E9B5010000 <1>      jmp     K26A ; THROW AWAY THIS CODE
715      <1>      ;
716      <1>      NOT_HC:
717 000011BF 247F      <1>      and     al, 07Fh ; TURN OFF THE BREAK BIT
718 000011C1 F6C702     <1>      test    bh, LC_E0 ; LAST CODE THE E0 MARKER CODE
719 000011C4 7410      <1>      jz      short NOT_LC_E0 ; JUMP IF NOT
720      <1>      ;
721 000011C6 BF[D6660000] <1>      mov     edi, _K6+6 ; IS THIS A SHIFT KEY?
722 000011CB AE        <1>      scasb ;
723      <1>      ; je     K26 ; K16B ; YES, THROW AWAY & RESET FLAG
724      <1>      ; 12/04/2021
725 000011CC 745B      <1>      je     short K16B ; K26
726 000011CE AE        <1>      scasb ;
727 000011CF 756D      <1>      jne     short K16A ; NO, CONTINUE KEY PROCESSING
728      <1>      ; jmp     short K16B ; YES, THROW AWAY & RESET FLAG
729 000011D1 E997010000 <1>      jmp     K26
730      <1>      ;
731      <1>      NOT_LC_E0:
732 000011D6 F6C701     <1>      test    bh, LC_E1 ; LAST CODE THE E1 MARKER CODE?
733 000011D9 7425      <1>      jz      short T_SYS_KEY ; JUMP IF NOT
734 000011DB B904000000 <1>      mov     ecx, 4 ; LENGHT OF SEARCH
735 000011E0 BF[D4660000] <1>      mov     edi, _K6+4 ; IS THIS AN ALT, CTL, OR SHIFT?
736 000011E5 F2AE      <1>      repne   scasb ; CHECK IT
737 000011E7 74D1      <1>      je     short EXIT ; THROW AWAY IF SO
738      <1>      ; 12/04/2021
739      <1>      ; je     K26A
740      <1>      ;
741 000011E9 3C45      <1>      cmp     al, NUM_KEY ; IS IT THE PAUSE KEY?
742 000011EB 753C      <1>      jne     short K16B ; NO, THROW AWAY & RESET FLAG
743      <1>      ; 12/04/2021
744      <1>      ; jmp     K26
745 000011ED F6C480     <1>      test    ah, 80h ; YES, IS IT THE BREAK OF THE KEY?
746 000011F0 7537      <1>      jnz     short K16B ; YES, THROW THIS AWAY, TOO
747      <1>      ; 24/07/2022
748      <1>      ; jnz     K26

```

```

749                                     <1> ; 20/02/2015
750 000011F2 F605[EA670000]08 <1> test byte [KB_FLAG_1],HOLD_STATE ; NO, ARE WE PAUSED ALREADY?
751 000011F9 752E <1> jnz short K16B ; YES, THROW AWAY
752 <1> ; 12/04/2021
753 <1> ;jnz K26
754 000011FB E9B1020000 <1> jmp K39P ; NO, THIS IS THE REAL PAUSE STATE
755 <1>
756 <1> ;----- TEST FOR SYSTEM KEY
757 <1> T_SYS_KEY:
758 00001200 3C54 <1> cmp al, SYS_KEY ; IS IT THE SYSTEM KEY?
759 00001202 753A <1> jnz short K16A ; CONTINUE IF NOT
760 <1>
761 00001204 F6C480 <1> test ah, 80h ; CHECK IF THIS A BREAK CODE
762 00001207 7525 <1> jnz short K16C ; DO NOT TOUCH SYSTEM INDICATOR IF TRUE
763 <1>
764 00001209 F605[EA670000]04 <1> test byte [KB_FLAG_1], SYS_SHIFT ; SEE IF IN SYSTEM KEY HELD DOWN
765 00001210 7517 <1> jnz short K16B ; IF YES, DO NOT PROCESS SYSTEM INDICATOR
766 <1> ; 12/04/2021
767 <1> ;jnz K26
768 <1>
769 00001212 800D[EA670000]04 <1> or byte [KB_FLAG_1], SYS_SHIFT ; INDICATE SYSTEM KEY DEPRESSED
770 00001219 B020 <1> mov al, EOI ; END OF INTERRUPT COMMAND
771 0000121B E620 <1> out 20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
772 <1> ; INTERRUPT-RETURN-NO-EOI
773 0000121D B0AE <1> mov al, ENA_KBD ; INSURE KEYBOARD IS ENABLED
774 0000121F E82D040000 <1> call SHIP_IT ; EXECUTE ENABLE
775 <1> ; !!! SYSREQ !!! function/system call (INTERRUPT) must be here !!!
776 <1> ;MOV AL, 8500H ; FUNCTION VALUE FOR MAKE OF SYSTEM KEY
777 <1> ;STI ; MAKE SURE INTERRUPTS ENABLED
778 <1> ;INT 15H ; USER INTERRUPT
779 00001224 E957010000 <1> jmp K27A ; END PROCESSING
780 <1>
781 00001229 E93F010000 <1> K16B: jmp K26 ; IGNORE SYSTEM KEY
782 <1>
783 <1> K16C:
784 0000122E 8025[EA670000]FB <1> and byte [KB_FLAG_1], ~SYS_SHIFT ; TURN OFF SHIFT KEY HELD DOWN
785 00001235 B020 <1> mov al, EOI ; END OF INTERRUPT COMMAND
786 00001237 E620 <1> out 20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
787 <1> ; INTERRUPT-RETURN-NO-EOI
788 <1> ;MOV AL, ENA_KBD ; INSURE KEYBOARD IS ENABLED
789 <1> ;CALL SHIP_IT ; EXECUTE ENABLE
790 <1>
791 <1> ;MOV AX, 8501H ; FUNCTION VALUE FOR BREAK OF SYSTEM KEY
792 <1> ;STI ; MAKE SURE INTERRUPTS ENABLED
793 <1> ;INT 15H ; USER INTERRUPT
794 <1> ;JMP K27A ; INCONRE SYSTEM KEY
795 <1>
796 00001239 E93B010000 <1> jmp K27 ; IGNORE SYSTEM KEY
797 <1>
798 <1> ;----- TEST FOR SHIFT KEYS
799 <1> K16A:
800 0000123E 8A1D[E9670000] <1> mov bl, [KB_FLAG] ; PUT STATE FLAGS IN BL
801 00001244 BF[D0660000] <1> mov edi, _K6 ; SHIFT KEY TABLE offset
802 00001249 B908000000 <1> mov ecx, _K6L ; LENGTH
803 0000124E F2AE <1> repne scasb ; LOOK THROUGH THE TABLE FOR A MATCH
804 00001250 88E0 <1> mov al, ah ; RECOVER SCAN CODE
805 <1> ;jne K25 ; IF NO MATCH, THEN SHIFT NOT FOUND
806 <1> ; 12/04/2021
807 00001252 7405 <1> je short K17
808 00001254 E9FC000000 <1> jmp K25
809 <1>
810 <1> ;----- SHIFT KEY FOUND
811 <1> K17:
812 00001259 81EF[D1660000] <1> sub edi, _K6+1 ; ADJUST PTR TO SCAN CODE MATCH
813 0000125F 8AA7[D8660000] <1> mov mov ah, [edi+_K7] ; GET MASK INTO AH
814 00001265 B102 <1> mov cl, 2 ; SETUP COUNT FOR FLAG SHIFTS
815 00001267 A880 <1> test al, 80h ; TEST FOR BREAK KEY
816 <1> ;jnz short K23 ; JUMP OF BREAK
817 <1> ; 12/04/2021
818 00001269 7405 <1> jz short K17C
819 0000126B E981000000 <1> jmp K23
820 <1>
821 <1> ;----- SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
822 <1> K17C:
823 00001270 80FC10 <1> cmp ah, SCROLL_SHIFT
824 00001273 7324 <1> jae short K18 ; IF SCROLL SHIFT OR ABOVE, TOGGLE KEY
825 <1>
826 <1> ;----- PLAIN SHIFT KEY, SET SHIFT ON
827 00001275 0825[E9670000] <1> or [KB_FLAG], ah ; TURN ON SHIFT BIT
828 0000127B A80C <1> test al, CTL_SHIFT+ALT_SHIFT ; IS IT ALT OR CTRL?
829 <1> ;jnz short K17D ; YES, MORE FLAGS TO SET
830 <1> ;jz K26 ; NO, INTERRUPT RETURN
831 <1> ; 12/04/2021
832 0000127D 7415 <1> jz short K17f
833 <1> K17D:
834 0000127F F6C702 <1> test bh, LC_E0 ; IS THIS ONE OF NEW KEYS?
835 00001282 7408 <1> jz short K17E ; NO, JUMP
836 00001284 0825[EC670000] <1> or [KB_FLAG_3], ah ; SET BITS FOR RIGHT CTRL, ALT
837 <1> ;jmp K26 ; INTERRUPT RETURN
838 <1> ; 12/04/2021
839 0000128A EB08 <1> jmp short K17f
840 <1> K17E:
841 0000128C D2EC <1> shr ah, cl ; MOVE FLAG BITS TWO POSITIONS
842 0000128E 0825[EA670000] <1> or [KB_FLAG_1], ah ; SET BITS FOR LEFT CTRL, ALT
843 <1> K17f: ; 12/04/2021
844 00001294 E9D4000000 <1> jmp K26
845 <1>
846 <1> ;----- TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
847 <1> K18: ; SHIFT-TOGGLE
848 00001299 F6C304 <1> test bl, CTL_SHIFT ; CHECK CTL SHIFT STATE
849 0000129C 7402 <1> jz short K18A ; JUMP IF NOT CTL STATE
850 <1> ;jnz K25 ; JUMP IF CTL STATE
851 <1> ; 12/04/2021
852 0000129E EB1C <1> jmp short K20a ; K25
853 <1> K18A:
854 000012A0 3C52 <1> cmp al, INS_KEY ; CHECK FOR INSERT KEY
855 000012A2 7522 <1> jne short K22 ; JUMP IF NOT INSERT KEY
856 000012A4 F6C308 <1> test bl, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
857 000012A7 7402 <1> jz short K18B ; JUMP IF NOT ALTERNATE SHIFT
858 <1> ;jnz K25 ; JUMP IF ALTERNATE SHIFT
859 <1> ; 12/04/2021
860 000012A9 EB11 <1> jmp short K20a ; K25
861 <1> K18B:
862 000012AB F6C702 <1> test bh, LC_E0 ; 20/02/2015 ; IS THIS NEW INSERT KEY?
863 000012AE 7516 <1> jnz short K22 ; YES, THIS ONE'S NEVER A '0'
864 <1> K19:
865 000012B0 F6C320 <1> test bl, NUM_STATE ; CHECK FOR BASE STATE
866 000012B3 750C <1> jnz short K21 ; JUMP IF NUM LOCK IS ON
867 000012B5 F6C303 <1> test bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT STATE
868 000012B8 740C <1> jz short K22 ; JUMP IF BASE STATE
869 <1> K20: ; NUMERIC ZERO, NOT INSERT KEY
870 000012BA 88C4 <1> mov ah, al ; PUT SCAN CODE BACK IN AH
871 <1> K20a: ; 12/04/2021
872 000012BC E994000000 <1> jmp K25 ; NUMERAL '0', STNDRD. PROCESSING

```

```

873      <1> K21:                                ; MIGHT BE NUMERIC
874 000012C1 F6C303      <1> test    b1, LEFT_SHIFT+RIGHT_SHIFT
875 000012C4 74F4      <1> jz      short K20                ; IS NUMERIC, STD. PROC.
876      <1> ;
877      <1> K22:                                ; SHIFT TOGGLE KEY HIT; PROCESS IT
878 000012C6 8425[EA670000] <1> test    ah, [KB_FLAG_1]                ; IS KEY ALREADY DEPRESSED
879      <1> ;jnz      short K26                ; JUMP IF KEY ALREADY DEPRESSED
880      <1> ; 12/04/2021
881 000012CC 75C6      <1> jnz      short k17f ; K26
882      <1> K22A:
883 000012CE 0825[EA670000] <1> or      [KB_FLAG_1], ah                ; INDICATE THAT THE KEY IS DEPRESSED
884 000012D4 3025[E9670000] <1> xor      [KB_FLAG], ah                ; TOGGLE THE SHIFT STATE
885      <1> ;
886      <1> ;----- TOGGLE LED IF CAPS, NUM OR SCROLL KEY DEPRESSED
887 000012DA F6C470      <1> test    ah, CAPS_SHIFT+NUM_SHIFT+SCROLL_SHIFT ; SHIFT TOGGLE?
888 000012DD 7407      <1> jz      short K22B                ; GO IF NOT
889      <1> ;
890      <1> ; 12/04/2021 (32 bit push/pop)
891 000012DF 50      <1> push    eax ; push ax                ; SAVE SCAN CODE AND SHIFT MASK
892 000012E0 E8C7030000 <1> call    SND_LED                ; GO TURN MODE INDICATORS ON
893 000012E5 58      <1> pop      eax ; pop ax                ; RESTORE SCAN CODE
894      <1> K22B:
895 000012E6 3C52      <1> cmp      al, INS_KEY                ; TEST FOR 1ST MAKE OF INSERT KEY
896      <1> ;jne      short K26                ; JUMP IF NOT INSERT KEY
897      <1> ; 12/04/2021
898 000012E8 75AA      <1> jne      short k17f ; K26
899 000012EA 88C4      <1> mov      ah, al                ; SCAN CODE IN BOTH HALVES OF AX
900 000012EC E999000000 <1> jmp      K28                ; FLAGS UPDATED, PROC. FOR BUFFER
901      <1> ;
902      <1> ;----- BREAK SHIFT FOUND
903      <1> K23:                                ; BREAK-SHIFT-FOUND
904 000012F1 80FC10      <1> cmp      ah, SCROLL_SHIFT                ; IS THIS A TOGGLE KEY
905 000012F4 F6D4      <1> not      ah                ; INVERT MASK
906 000012F6 7355      <1> jae      short K24                ; YES, HANDLE BREAK TOGGLE
907 000012F8 2025[E9670000] <1> and     [KB_FLAG], ah                ; TURN OFF SHIFT BIT
908 000012FE 80FCFB      <1> cmp      ah, ~CTL_SHIFT                ; IS THIS ALT OR CTL?
909 00001301 7730      <1> ja      short K23D                ; NO, ALL DONE
910      <1> ;
911 00001303 F6C702      <1> test    bh, LC_E0                ; 2ND ALT OR CTL?
912 00001306 7408      <1> jz      short K23A                ; NO, HANSLE NORMALLY
913 00001308 2025[EC670000] <1> and     [KB_FLAG_3], ah                ; RESET BIT FOR RIGHT ALT OR CTL
914 0000130E EB08      <1> jmp      short K23B                ; CONTINUE
915      <1> K23A:
916 00001310 D2FC      <1> sar      ah, cl                ; MOVE THE MASK BIT TWO POSITIONS
917 00001312 2025[EA670000] <1> and     [KB_FLAG_1], ah                ; RESET BIT FOR LEFT ALT AND CTL
918      <1> K23B:
919 00001318 88C4      <1> mov      ah, al                ; SAVE SCAN CODE
920 0000131A A0[EC670000] <1> mov      al, [KB_FLAG_3]                ; GET RIGHT ALT & CTRL FLAGS
921 0000131F D2E8      <1> shr      al, cl                ; MOVE TO BITS 1 & 0
922 00001321 0A05[EA670000] <1> or      al, [KB_FLAG_1]                ; PUT IN LEFT ALT & CTL FLAGS
923 00001327 D2E0      <1> shl      al, cl                ; MOVE BACK TO BITS 3 & 2
924 00001329 240C      <1> and     al, ALT_SHIFT+CTL_SHIFT        ; FILTER OUT OTHER GARBAGE
925 0000132B 0805[E9670000] <1> or      [KB_FLAG], al                ; PUT RESULT IN THE REAL FLAGS
926 00001331 88E0      <1> mov      al, ah
927      <1> K23D:
928 00001333 3CB8      <1> cmp      al, ALT_KEY+80h                ; IS THIS ALTERNATE SHIFT RELEASE
929 00001335 7536      <1> jne      short K26                ; INTERRUPT RETURN
930      <1> ;
931      <1> ;----- ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
932 00001337 A0[ED670000] <1> mov      al, [ALT_INPUT]
933 0000133C B400      <1> mov      ah, 0                ; SCAN CODE OF 0
934 0000133E 8825[ED670000] <1> mov      [ALT_INPUT], ah                ; ZERO OUT THE FIELD
935 00001344 3C00      <1> cmp      al, 0                ; WAS THE INPUT = 0?
936 00001346 7425      <1> je      short K26                ; INTERRUPT_RETURN
937      <1> ; 29/01/2016
938      <1> ;jmp      K61                ; IT WASN'T, SO PUT IN BUFFER
939 00001348 E9AB020000 <1> jmp      _K60
940      <1> ;
941      <1> K24:                                ; BREAK-TOGGLE
942 0000134D 2025[EA670000] <1> and     [KB_FLAG_1], ah                ; INDICATE NO LONGER DEPRESSED
943 00001353 EB18      <1> jmp      short K26                ; INTERRUPT_RETURN
944      <1> ;
945      <1> ;----- TEST FOR HOLD STATE
946      <1> ; AL, AH = SCAN CODE
947      <1> K25:                                ; NO-SHIFT-FOUND
948 00001355 3C80      <1> cmp      al, 80h                ; TEST FOR BREAK KEY
949 00001357 7314      <1> jae      short K26                ; NOTHING FOR BREAK CHARS FROM HERE ON
950 00001359 F605[EA670000]08 <1> test    byte [KB_FLAG_1], HOLD_STATE ; ARE WE IN HOLD STATE
951 00001360 7428      <1> jz      short K28                ; BRANCH AROUND TEST IF NOT
952 00001362 3C45      <1> cmp      al, NUM_KEY                ; CAN'T END HOLD ON NUM_LOCK
953 00001364 7407      <1> je      short K26                ; TURN OFF THE HOLD STATE BIT
954 00001366 8025[EA670000]F7 <1> and     byte [KB_FLAG_1], ~HOLD_STATE
955      <1> K26:
956 0000136D 8025[EC670000]FC <1> and     byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
957      <1> K26A:
958 00001374 FA      <1> cli                ; TURN OFF INTERRUPTS
959 00001375 B020      <1> mov      al, EOI                ; END OF INTERRUPT COMMAND
960 00001377 E620      <1> out      20h, al ;out INTA00, al        ; SEND COMMAND TO INTERRUPT CONTROL PORT
961      <1> K27:
962 00001379 B0AE      <1> mov      al, ENA_KBD                ; INSURE KEYBOARD IS ENABLED
963 0000137B E8D1020000 <1> call    SHIP_IT                ; EXECUTE ENABLE
964      <1> K27A:
965 00001380 FA      <1> cli                ; DISABLE INTERRUPTS
966      <1> ;mov     byte [intflg], 0 ; 07/01/2017 ; 15/01/2017
967 00001381 07      <1> pop      es                ; RESTORE REGISTERS
968 00001382 1F      <1> pop      ds
969 00001383 5F      <1> pop      edi
970 00001384 5E      <1> pop      esi
971 00001385 5A      <1> pop      edx
972 00001386 59      <1> pop      ecx
973 00001387 5B      <1> pop      ebx
974 00001388 58      <1> pop      eax
975      <1> ;pop     ebp
976 00001389 CF      <1> iretd                ; RETURN
977      <1> ;
978      <1> ;----- NOT IN HOLD STATE
979      <1> K28:                                ; NO-HOLD-STATE
980 0000138A 3C58      <1> cmp      al, 88                ; TEST FOR OUT-OF-RANGE SCAN CODES
981 0000138C 77DF      <1> ja      short K26                ; IGNORE IF OUT-OF-RANGE
982      <1> ;
983 0000138E F6C308      <1> test    b1, ALT_SHIFT                ; ARE WE IN ALTERNATE SHIFT
984 00001391 740E      <1> jz      short K28A                ; IF NOT ALTERNATE
985      <1> ; 12/04/2021
986      <1> ;jz      K38
987      <1> ;
988 00001393 F6C710      <1> test    bh, KBX                ; IS THIS THE ENCHANCED KEYBOARD?
989 00001396 740E      <1> jz      short K29                ; NO, ALT STATE IS REAL
990      <1> ;28/02/2015
991 00001398 F605[EA670000]04 <1> test    byte [KB_FLAG_1], SYS_SHIFT ; YES, IS SYSREQ KEY DOWN?
992 0000139F 7405      <1> jz      short K29                ; NO, ALT STATE IS REAL
993      <1> ; 12/04/2021
994      <1> ;jnz     K38
995      <1> ;
996 000013A1 E9C4000000 <1> K28A:    jmp      K38

```

```

997      <1>      ;
998      <1>      ;----- TEST FOR RESET KEY SEQUENCE (CTL ALT DEL)
999      <1>      ;
1000     000013A6 F6C304      <1>      K29:      ; TEST-RESET
1001     000013A9 740B      <1>      test    bl, CTL_SHIFT      ; ARE WE IN CONTROL SHIFT ALSO?
1002     000013AB 3C53      <1>      jz      short K31      ; NO_RESET
1003     000013AD 7507      <1>      cmp     al, DEL_KEY      ; CTL-ALT STATE, TEST FOR DELETE KEY
1004     <1>      jne     short K31      ; NO_RESET, IGNORE
1005     <1>      ;
1006     <1>      ;----- CTL-ALT-DEL HAS BEEN FOUND
1007     <1>      ; 26/08/2014
1008     <1>      cpu_reset:
1009     <1>      ; IBM PC/AT ROM BIOS source code - 10/06/85 (TEST4.ASM - PROC_SHUTDOWN)
1010     000013AF B0FE      <1>      ; Send FEh (system reset command) to the keyboard controller.
1011     000013B1 E664      <1>      mov     al, SHUT_CMD      ; SHUTDOWN COMMAND
1012     <1>      out     STATUS_PORT, al      ; SEND TO KEYBOARD CONTROL PORT
1013     000013B3 F4      <1>      khere:      hlt      ; WAIT FOR 80286 RESET
1014     000013B4 EBF0      <1>      jmp     short khere      ; INSURE HALT
1015     <1>      ;
1016     <1>      ;----- IN ALTERNATE SHIFT, RESET NOT FOUND
1017     <1>      K31:      ; NO-RESET
1018     000013B6 3C39      <1>      cmp     al, 57      ; TEST FOR SPACE KEY
1019     000013B8 7507      <1>      jne     short K311      ; NOT THERE
1020     000013BA B020      <1>      mov     al, ' '      ; SET SPACE CHAR
1021     <1>      K31a:      ; 12/04/2021
1022     000013BC E929020000      <1>      jmp     K57      ; BUFFER_FILL
1023     <1>      K311:
1024     000013C1 3C0F      <1>      cmp     al, 15      ; TEST FOR TAB KEY
1025     000013C3 7506      <1>      jne     short K312      ; NOT THERE
1026     000013C5 66B800A5      <1>      mov     ax, 0A500h      ; SET SPECIAL CODE FOR ALT-TAB
1027     <1>      ; jmp     K57      ; BUFFER_FILL
1028     <1>      ; 12/04/2021
1029     000013C9 EBF1      <1>      jmp     short K31a
1030     <1>      K312:
1031     000013CB 3C4A      <1>      cmp     al, 74      ; TEST FOR KEY PAD -
1032     <1>      ; je short K37B      ; GO PROCESS
1033     <1>      ; 12/04/2021
1034     000013CD 7404      <1>      je     short K312a
1035     000013CF 3C4E      <1>      cmp     al, 78      ; TEST FOR KEY PAD +
1036     <1>      ; je short K37B      ; GO PROCESS
1037     <1>      ; 12/04/2021
1038     000013D1 7505      <1>      jne     short K32
1039     <1>      K312a:
1040     000013D3 E988000000      <1>      jmp     K37B
1041     <1>      ;
1042     <1>      ;----- LOOK FOR KEY PAD ENTRY
1043     <1>      K32:
1044     000013D8 BFAC660000      <1>      mov     edi, K30      ; ALT-KEY-PAD
1045     000013DD B90A000000      <1>      mov     ecx, 10      ; ALT-INPUT-TABLE offset
1046     000013E2 F2AE      <1>      repne   scasb      ; LOOK FOR ENTRY USING KEYPAD
1047     000013E4 7521      <1>      jne     short K33      ; LOOK FOR MATCH
1048     000013E6 F6C702      <1>      jne     short K33      ; NO-ALT-KEYPAD
1049     000013E9 7579      <1>      test    bh, LC_E0      ; IS THIS ONE OF THE NEW KEYS?
1050     000013EB 81EFAD660000      <1>      jnz     short K37C      ; YES, JUMP, NOT NUMPAD KEY
1051     000013F1 A0ED670000      <1>      sub     edi, K30+1      ; DI NOW HAS ENTRY VALUE
1052     000013F6 B40A      <1>      mov     al, [ALT_INPUT]      ; GET THE CURRENT BYTE
1053     000013F8 F6E4      <1>      mov     ah, 10      ; MULTIPLY BY 10
1054     000013FA 6601F8      <1>      mul     ah
1055     000013FD A2ED670000      <1>      add     ax, di      ; ADD IN THE LATEST ENTRY
1056     <1>      mov     [ALT_INPUT], al      ; STORE IT AWAY
1057     00001402 E966FFFFFF      <1>      K32A:      jmp     K26      ; THROW AWAY THAT KEYSTROKE
1058     <1>      ;
1059     <1>      ;----- LOOK FOR SUPERSHIFT ENTRY
1060     <1>      K33:      ; NO-ALT-KEYPAD
1061     00001407 C605ED670000      <1>      mov     byte [ALT_INPUT], 0      ; ZERO ANY PREVIOUS ENTRY INTO INPUT
1062     0000140E B91A000000      <1>      mov     ecx, 26      ; (DI),(ES) ALREADY POINTING
1063     00001413 F2AE      <1>      repne   scasb      ; LOOK FOR MATCH IN ALPHABET
1064     00001415 7445      <1>      je     short K37A      ; MATCH FOUND, GO FILL THE BUFFER
1065     <1>      ;
1066     <1>      ;----- LOOK FOR TOP ROW OF ALTERNATE SHIFT
1067     <1>      K34:      ; ALT-TOP-ROW
1068     00001417 3C02      <1>      cmp     al, 2      ; KEY WITH '1' ON IT
1069     00001419 7245      <1>      jb     short K37B      ; MUST BE ESCAPE
1070     0000141B 3C0D      <1>      cmp     al, 13      ; IS IT IN THE REGION
1071     0000141D 7705      <1>      ja     short K35      ; NO, ALT SOMETHING ELSE
1072     0000141F 80C476      <1>      add     ah, 118      ; CONVERT PSEUDO SCAN CODE TO RANGE
1073     00001422 EB38      <1>      jmp     short K37A      ; GO FILL THE BUFFER
1074     <1>      ;
1075     <1>      ;----- TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
1076     <1>      K35:      ; ALT-FUNCTION
1077     00001424 3C57      <1>      cmp     al, F11_M      ; IS IT F11?
1078     00001426 7209      <1>      jb     short K35A ; 20/02/2015      ; NO, BRANCH
1079     00001428 3C58      <1>      cmp     al, F12_M      ; IS IT F12?
1080     0000142A 7705      <1>      ja     short K35A ; 20/02/2015      ; NO, BRANCH
1081     0000142C 80C434      <1>      add     ah, 52      ; CONVERT TO PSEUDO SCAN CODE
1082     0000142F EB2B      <1>      jmp     short K37A      ; GO FILL THE BUFFER
1083     <1>      K35A:
1084     00001431 F6C702      <1>      test    bh, LC_E0      ; DO WE HAVE ONE OF THE NEW KEYS?
1085     00001434 741B      <1>      jz     short K37      ; NO, JUMP
1086     00001436 3C1C      <1>      cmp     al, 28      ; TEST FOR KEYPAD ENTER
1087     00001438 7506      <1>      jne     short K35B      ; NOT THERE
1088     0000143A 66B800A6      <1>      mov     ax, 0A600h      ; SPECIAL CODE
1089     <1>      ; jmp     K57      ; BUFFER FILL
1090     <1>      ; 12/04/2021
1091     0000143E EB0C      <1>      jmp     short K35c
1092     <1>      K35B:
1093     00001440 3C53      <1>      cmp     al, 83      ; TEST FOR DELETE KEY
1094     00001442 7420      <1>      je     short K37C      ; HANDLE WITH OTHER EDIT KEYS
1095     00001444 3C35      <1>      cmp     al, 53      ; TEST FOR KEYPAD /
1096     00001446 75BA      <1>      jne     short K32A      ; NOT THERE, NO OTHER E0 SPECIALS
1097     <1>      ; 12/04/2021
1098     <1>      ; jne     K26
1099     00001448 66B800A4      <1>      mov     ax, 0A400h      ; SPECIAL CODE1
1100     <1>      K35c:      ; 12/04/2021
1101     0000144C E999010000      <1>      jmp     K57      ; BUFFER FILL
1102     <1>      K37:
1103     00001451 3C3B      <1>      cmp     al, 59      ; TEST FOR FUNCTION KEYS (F1)
1104     00001453 720B      <1>      jb     short K37B      ; NO FN, HANDLE W/OTHER EXTENDED
1105     00001455 3C44      <1>      cmp     al, 68      ; IN KEYPAD REGION?
1106     00001457 77A9      <1>      ja     short K32A      ; IF SO, IGNORE
1107     <1>      ; 12/04/2021
1108     <1>      ; ja     K26
1109     00001459 80C42D      <1>      add     ah, 45      ; CONVERT TO PSEUDO SCAN CODE
1110     <1>      K37A:
1111     0000145C B000      <1>      mov     al, 0      ; ASCII CODE OF ZERO
1112     <1>      ; jmp     K57      ; PUT IT IN THE BUFFER
1113     <1>      ; 12/04/2021
1114     0000145E EBEC      <1>      jmp     short K35c
1115     <1>      K37B:
1116     00001460 B0F0      <1>      mov     al, 0F0h      ; USE SPECIAL ASCII CODE
1117     <1>      ; jmp     K57      ; PUT IT IN THE BUFFER
1118     <1>      ; 12/04/2021
1119     00001462 EBE8      <1>      jmp     short K35c
1120     <1>      K37C:

```

```

1121 00001464 0450      <1>      add     al, 80          ; CONVERT SCAN CODE (EDIT KEYS)
1122 00001466 88C4      <1>      mov     ah, al          ; (SCAN CODE NOT IN AH FOR INSERT)
1123 00001468 EBF2      <1>      jmp     short K37A        ; PUT IT IN THE BUFFER
1124                                <1>      ;
1125                                <1>      ;----- NOT IN ALTERNATE SHIFT
1126                                <1>      ;
1127                                <1>      K38:          ; NOT-ALT-SHIFT
1128 0000146A F6C304     <1>      test    bl, CTL_SHIFT      ; BL STILL HAS SHIFT FLAGS
1129                                <1>      ;jnz     short K38A        ; ARE WE IN CONTROL SHIFT?
1130                                <1>      ;jz      K44          ; YES, START PROCESSING
1131                                <1>      ; 12/04/2021        ; NOT-CTL-SHIFT
1132 0000146D 7505      <1>      jnz     short K38A        ; YES, START PROCESSING
1133 0000146F E9AB000000 <1>      jmp     K44          ; NOT-CTL-SHIFT
1134                                <1>      ;
1135                                <1>      ;----- CONTROL SHIFT, TEST SPECIAL CHARACTERS
1136                                <1>      ;----- TEST FOR BREAK
1137                                <1>      K38A:          ;
1138 00001474 3C46      <1>      cmp     al, SCROLL_KEY      ; TEST FOR BREAK
1139 00001476 7530      <1>      jne     short K39          ; JUMP, NO-BREAK
1140 00001478 F6C710     <1>      test    bh, KBX          ; IS THIS THE ENHANCED KEYBOARD?
1141 0000147B 7405      <1>      jz      short K38B        ; NO, BREAK IS VALID
1142 0000147D F6C702     <1>      test    bh, LC_E0        ; YES, WAS LAST CODE AN E0?
1143 00001480 7426      <1>      jz      short K39          ; NO-BREAK, TEST FOR PAUSE
1144                                <1>      K38B:          ;
1145 00001482 8B1D[F6670000] <1>      mov     ebx, [BUFFER_HEAD] ; RESET BUFFER TO EMPTY
1146 00001488 891D[FA670000] <1>      mov     [BUFFER_TAIL], ebx
1147 0000148E C605[E8670000]80 <1>      mov     byte [BIOS_BREAK], 80h ; TURN ON BIOS_BREAK BIT
1148                                <1>      ;
1149                                <1>      ;----- ENABLE KEYBOARD
1150 00001495 B0AE      <1>      mov     al, ENA_KBD          ; ENABLE KEYBOARD
1151 00001497 E8B5010000 <1>      call    SHIP_IT          ; EXECUTE ENABLE
1152                                <1>      ;
1153                                <1>      ; CTRL+BREAK code here !!!
1154                                <1>      ;INT     1BH          ; BREAK INTERRUPT VECTOR
1155                                <1>      ; 17/10/2015
1156 0000149C E88D590000 <1>      call    ctrlbrk ; control+break subroutine
1157                                <1>      ;
1158                                <1>      ;sub     ax, ax          ; PUT OUT DUMMY CHARACTER
1159                                <1>      ; 12/04/2021
1160 000014A1 29C0      <1>      sub     eax, eax
1161 000014A3 E942010000 <1>      jmp     K57          ; BUFFER_FILL
1162                                <1>      ;
1163                                <1>      ;----- TEST FOR PAUSE
1164                                <1>      K39:          ; NO_BREAK
1165 000014A8 F6C710     <1>      test    bh, KBX          ; IS THIS THE ENHANCED KEYBOARD?
1166 000014AB 7537      <1>      jnz     short K41          ; YES, THEN THIS CAN'T BE PAUSE
1167 000014AD 3C45      <1>      cmp     al, NUM_KEY        ; LOOK FOR PAUSE KEY
1168 000014AF 7533      <1>      jne     short K41          ; NO-PAUSE
1169                                <1>      K39P:          ;
1170 000014B1 800D[EA670000]08 <1>      or      byte [KB_FLAG_1], HOLD_STATE ; TURN ON THE HOLD FLAG
1171                                <1>      ;
1172                                <1>      ;----- ENABLE KEYBOARD
1173 000014B8 B0AE      <1>      mov     al, ENA_KBD          ; ENABLE KEYBOARD
1174 000014BA E892010000 <1>      call    SHIP_IT          ; EXECUTE ENABLE
1175                                <1>      K39A:          ;
1176 000014BF B020      <1>      mov     al, EOI          ; END OF INTERRUPT TO CONTROL PORT
1177 000014C1 E620      <1>      out     20h, al ;out INTA00, al ; ALLOW FURTHER KEYSTROKE INTERRUPTS
1178                                <1>      ;
1179                                <1>      ;----- DURING PAUSE INTERVAL, TURN COLOR CRT BACK ON
1180 000014C3 803D[1E680000]07 <1>      cmp     byte [CRT_MODE], 7 ; IS THIS BLACK AND WHITE CARD
1181 000014CA 740A      <1>      je      short K40          ; YES, NOTHING TO DO
1182 000014CC 66BAD803 <1>      mov     dx, 03D8h          ; PORT FOR COLOR CARD
1183 000014D0 A0[1F680000] <1>      mov     al, [CRT_MODE_SET] ; GET THE VALUE OF THE CURRENT MODE
1184 000014D5 EE        <1>      out     dx, al          ; SET THE CRT MODE, SO THAT CRT IS ON
1185                                <1>      ;
1186                                <1>      K40:          ; PAUSE-LOOP
1187 000014D6 F605[EA670000]08 <1>      test    byte [KB_FLAG_1], HOLD_STATE ; CHECK HOLD STATE FLAG
1188 000014DD 75F7      <1>      jnz     short K40          ; LOOP UNTIL FLAG TURNED OFF
1189                                <1>      ;
1190 000014DF E995FEFFFF <1>      jmp     K27          ; INTERRUPT_RETURN_NO_EOI
1191                                <1>      ;
1192                                <1>      ;----- TEST SPECIAL CASE KEY 55
1193                                <1>      K41:          ; NO-PAUSE
1194 000014E4 3C37      <1>      cmp     al, 55          ; TEST FOR */PRTSC KEY
1195 000014E6 7513      <1>      jne     short K42          ; NOT-KEY-55
1196 000014E8 F6C710     <1>      test    bh, KBX          ; IS THIS THE ENHANCED KEYBOARD?
1197 000014EB 7405      <1>      jz      short K41A        ; NO, CTL-PRTSC IS VALID
1198 000014ED F6C702     <1>      test    bh, LC_E0        ; YES, WAS LAST CODE AN E0?
1199 000014F0 7421      <1>      jz      short K42B        ; NO, TRANSLATE TO A FUNCTION
1200                                <1>      K41A:          ;
1201 000014F2 66B80072 <1>      mov     ax, 114*256        ; START/STOP PRINTING SWITCH
1202 000014F6 E9EF000000 <1>      jmp     K57          ; BUFFER_FILL
1203                                <1>      ;
1204                                <1>      ;----- SET UP TO TRANSLATE CONTROL SHIFT
1205                                <1>      K42:          ; NOT-KEY-55
1206 000014FB 3C0F      <1>      cmp     al, 15          ; IS IT THE TAB KEY?
1207 000014FD 7414      <1>      je      short K42B        ; YES, XLATE TO FUNCTION CODE
1208 000014FF 3C35      <1>      cmp     al, 53          ; IS IT THE / KEY?
1209 00001501 750E      <1>      jne     short K42A        ; NO, NO MORE SPECIAL CASES
1210 00001503 F6C702     <1>      test    bh, LC_E0        ; YES, IS IT FROM THE KEY PAD?
1211 00001506 7409      <1>      jz      short K42A        ; NO, JUST TRANSLATE
1212 00001508 66B80095 <1>      mov     ax, 9500h          ; YES, SPECIAL CODE FOR THIS ONE
1213 0000150C E9D9000000 <1>      jmp     K57          ; BUFFER_FILL
1214                                <1>      K42A:          ;
1215                                <1>      ;mov     ebx, _K8          ; SET UP TO TRANSLATE CTL
1216 00001511 3C3B      <1>      cmp     al, 59          ; IS IT IN CHARACTER TABLE?
1217                                <1>      ;jb     short K45F        ; YES, GO TRANSLATE CHAR
1218                                <1>      ;;;jb    K56 ; 20/02/2015
1219                                <1>      ;;;jmp    K64 ; 20/02/2015
1220                                <1>      K42B:          ;
1221 00001513 BB[E0660000] <1>      mov     ebx, _K8          ; SET UP TO TRANSLATE CTL
1222                                <1>      ;jb     K56 ; 20/02/2015
1223                                <1>      ; 12/04/2021
1224 00001518 7267      <1>      jnb     short K45F
1225 0000151A E9B9000000 <1>      jmp     K64
1226                                <1>      ;
1227                                <1>      ;----- NOT IN CONTROL SHIFT
1228                                <1>      K44:          ; NOT-CTL-SHIFT
1229 0000151F 3C37      <1>      cmp     al, 55          ; PRINT SCREEN KEY?
1230 00001521 7528      <1>      jne     short K45          ; NOT PRINT SCREEN
1231 00001523 F6C710     <1>      test    bh, KBX          ; IS THIS ENHANCED KEYBOARD?
1232 00001526 7407      <1>      jz      short K44A        ; NO, TEST FOR SHIFT STATE
1233 00001528 F6C702     <1>      test    bh, LC_E0        ; YES, LAST CODE A MARKER?
1234 0000152B 7507      <1>      jnz     short K44B        ; YES, IS PRINT SCREEN
1235 0000152D EB41      <1>      jmp     short K45C        ; NO, TRANSLATE TO '*' CHARACTER
1236                                <1>      K44A:          ;
1237 0000152F F6C303     <1>      test    bl, LEFT_SHIFT+RIGHT_SHIFT ; NOT 101 KBD, SHIFT KEY DOWN?
1238 00001532 743C      <1>      jz      short K45C        ; NO, TRANSLATE TO '*' CHARACTER
1239                                <1>      ;
1240                                <1>      ;----- ISSUE INTERRUPT TO INDICATE PRINT SCREEN FUNCTION
1241                                <1>      K44B:          ;
1242 00001534 B0AE      <1>      mov     al, ENA_KBD          ; INSURE KEYBOARD IS ENABLED
1243 00001536 E816010000 <1>      call    SHIP_IT          ; EXECUTE ENABLE
1244 0000153B B020      <1>      mov     al, EOI          ; END OF CURRENT INTERRUPT

```

```

1245 0000153D E620      <1>      out      20h, al ;out INTA00, al      ; SO FURTHER THINGS CAN HAPPEN
1246                  <1>      ; Print Screen !!!      ; ISSUE PRINT SCREEN INTERRUPT (INT 05h)
1247                  <1>      ;PUSH BP      ; SAVE POINTER
1248                  <1>      ;INT 5H      ; ISSUE PRINT SCREEN INTERRUPT
1249                  <1>      ;POP BP      ; RESTORE POINTER
1250 0000153F 8025[EC670000]FC <1>      and      byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; ZERO OUT THESE FLAGS
1251 00001546 E92EFEFFFF <1>      jmp      K27      ; GO BACK WITHOUT EOI OCCURRING
1252                  <1>      ;
1253                  <1>      ;----- HANDLE IN-CORE KEYS
1254                  <1>      K45:      ; NOT-PRINT-SCREEN
1255 0000154B 3C3A      <1>      cmp      al, 58      ; TEST FOR IN-CORE AREA
1256 0000154D 7734      <1>      ja      short K46      ; JUMP IF NOT
1257 0000154F 3C35      <1>      cmp      al, 53      ; IS THIS THE '/' KEY?
1258 00001551 7505      <1>      jne      short K45A      ; NO, JUMP
1259 00001553 F6C702    <1>      test     bh, LC_E0      ; WAS THE LAST CODE THE MARKER?
1260 00001556 7518      <1>      jnz      short K45C      ; YES, TRANSLATE TO CHARACTER
1261                  <1>      K45A:
1262 00001558 B91A000000    <1>      mov      ecx, 26      ; LENGHT OF SEARCH
1263 0000155D BF[B6660000] <1>      mov      edi, K30+10    ; POINT TO TABLE OF A-Z CHARS
1264 00001562 F2AE      <1>      repne    scasb      ; IS THIS A LETTER KEY?
1265                  <1>      ; 20/02/2015
1266 00001564 7505      <1>      jne      short K45B      ; NO, SYMBOL KEY
1267                  <1>      ;
1268 00001566 F6C340    <1>      test     bl, CAPS_STATE ; ARE WE IN CAPS_LOCK?
1269 00001569 750C      <1>      jnz      short K45D      ; TEST FOR SURE
1270                  <1>      K45B:
1271 0000156B F6C303    <1>      test     bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
1272 0000156E 750C      <1>      jnz      short K45E      ; YES, UPPERCASE
1273                  <1>      ; NO, LOWERCASE
1274                  <1>      K45C:
1275 00001570 BB[38670000] <1>      mov      ebx, K10      ; TRANSLATE TO LOWERCASE LETTERS
1276 00001575 EB51      <1>      jmp      short K56
1277                  <1>      K45D:      ; ALMOST-CAPS-STATE
1278 00001577 F6C303    <1>      test     bl, LEFT_SHIFT+RIGHT_SHIFT ; CL ON. IS SHIFT ON, TOO?
1279 0000157A 75F4      <1>      jnz      short K45C      ; SHIFTED TEMP OUT OF CAPS STATE
1280                  <1>      K45E:
1281 0000157C BB[90670000] <1>      mov      ebx, K11      ; TRANSLATE TO UPPER CASE LETTERS
1282 00001581 EB45      <1>      jmp      short K56
1283                  <1>      ;
1284                  <1>      ;----- TEST FOR KEYS F1 - F10
1285                  <1>      K46:      ; NOT IN-CORE AREA
1286 00001583 3C44      <1>      cmp      al, 68      ; TEST FOR F1 - F10
1287                  <1>      ;ja      short K47      ; JUMP IF NOT
1288                  <1>      ;jmp     short K53      ; YES, GO DO FN KEY PROCESS
1289 00001585 7635      <1>      jna      short K53
1290                  <1>      ;
1291                  <1>      ;----- HANDLE THE NUMERIC PAD KEYS
1292                  <1>      K47:      ; NOT F1 - F10
1293 00001587 3C53      <1>      cmp      al, 83      ; TEST NUMPAD KEYS
1294 00001589 772D      <1>      ja      short K52      ; JUMP IF NOT
1295                  <1>      ;
1296                  <1>      ;----- KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
1297                  <1>      K48:
1298 0000158B 3C4A      <1>      cmp      al, 74      ; SPECIAL CASE FOR MINUS
1299 0000158D 74ED      <1>      je      short K45E      ; GO TRANSLATE
1300 0000158F 3C4E      <1>      cmp      al, 78      ; SPECIAL CASE FOR PLUS
1301 00001591 74E9      <1>      je      short K45E      ; GO TRANSLATE
1302 00001593 F6C702    <1>      test     bh, LC_E0      ; IS THIS ONE OFTHE NEW KEYS?
1303 00001596 750A      <1>      jnz      short K49      ; YES, TRANSLATE TO BASE STATE
1304                  <1>      ;
1305 00001598 F6C320    <1>      test     bl, NUM_STATE      ; ARE WE IN NUM LOCK
1306 0000159B 7514      <1>      jnz      short K50      ; TEST FOR SURE
1307 0000159D F6C303    <1>      test     bl, LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT STATE?
1308                  <1>      ;jnz     short K51      ; IF SHIFTED, REALLY NUM STATE
1309 000015A0 75DA      <1>      jnz      short K45E
1310                  <1>      ;
1311                  <1>      ;----- BASE CASE FOR KEYPAD
1312                  <1>      K49:
1313 000015A2 3C4C      <1>      cmp      al, 76      ; SPECIAL CASE FOR BASE STATE 5
1314 000015A4 7504      <1>      jne      short K49A      ; CONTINUE IF NOT KEYPAD 5
1315 000015A6 B0F0      <1>      mov      al, 0F0h      ; SPECIAL ASCII CODE
1316 000015A8 EB40      <1>      jmp      short K57      ; BUFFER FILL
1317                  <1>      K49A:
1318 000015AA BB[38670000] <1>      mov      ebx, K10      ; BASE CASE TABLE
1319 000015AF EB27      <1>      jmp      short K64      ; CONVERT TO PSEUDO SCAN
1320                  <1>      ;
1321                  <1>      ;----- MIGHT BE NUM LOCK, TEST SHIFT STATUS
1322                  <1>      K50:      ; ALMOST-NUM-STATE
1323 000015B1 F6C303    <1>      test     bl, LEFT_SHIFT+RIGHT_SHIFT
1324 000015B4 75EC      <1>      jnz      short K49      ; SHIFTED TEMP OUT OF NUM STATE
1325 000015B6 EBC4      <1>      jmp      short K45E      ; REALLY NUM STATE
1326                  <1>      ;
1327                  <1>      ;----- TEST FOR THE NEW KEYS ON WT KEYBOARDS
1328                  <1>      K52:      ; NOT A NUMPAD KEY
1329 000015B8 3C56      <1>      cmp      al, 86      ; IS IT THE NEW WT KEY?
1330                  <1>      ;jne     short K53      ; JUMP IF NOT
1331                  <1>      ;jmp     short K45B      ; HANDLE WITH REST OF LETTER KEYS
1332 000015BA 74AF      <1>      je      short K45B
1333                  <1>      ;
1334                  <1>      ;----- MUST BE F11 OR F12
1335                  <1>      K53:      ; F1 - F10 COME HERE, TOO
1336 000015BC F6C303    <1>      test     bl, LEFT_SHIFT+RIGHT_SHIFT ; TEST SHIFT STATE
1337 000015BF 74E1      <1>      jz      short K49      ; JUMP, LOWER CASE PSEUDO SC'S
1338                  <1>      ; 20/02/2015
1339 000015C1 BB[90670000] <1>      mov      ebx, K11      ; UPPER CASE PSEUDO SCAN CODES
1340 000015C6 EB10      <1>      jmp      short K64      ; TRANSLATE SCAN
1341                  <1>      ;
1342                  <1>      ;----- TRANSLATE THE CHARACTER
1343                  <1>      K56:      ; TRANSLATE-CHAR
1344 000015C8 FEC8      <1>      dec      al      ; CONVERT ORIGIN
1345 000015CA D7      <1>      xlat      ; CONVERT THE SCAN CODE TO ASCII
1346 000015CB F605[EC670000]02 <1>      test     byte [KB_FLAG_3], LC_E0 ; IS THIS A NEW KEY?
1347 000015D2 7416      <1>      jz      short K57      ; NO, GO FILL BUFFER
1348 000015D4 B4E0      <1>      mov      ah, MC_E0      ; YES, PUT SPECIAL MARKER IN AH
1349 000015D6 EB12      <1>      jmp      short K57      ; PUT IT INTO THE BUFFER
1350                  <1>      ;
1351                  <1>      ;----- TRANSLATE SCAN FOR PSEUDO SCAN CODES
1352                  <1>      K64:      ; TRANSLATE-SCAN-ORGD
1353 000015D8 FEC8      <1>      dec      al      ; CONVERT ORIGIN
1354 000015DA D7      <1>      xlat      ; CTL TABLE SCAN
1355 000015DB 88C4      <1>      mov      ah, al      ; PUT VALUE INTO AH
1356 000015DD B000      <1>      mov      al, 0      ; ZERO ASCII CODE
1357 000015DF F605[EC670000]02 <1>      test     byte [KB_FLAG_3], LC_E0 ; IS THIS A NEW KEY?
1358 000015E6 7402      <1>      jz      short K57      ; NO, GO FILL BUFFER
1359 000015E8 B0E0      <1>      mov      al, MC_E0      ; YES, PUT SPECIAL MARKER IN AL
1360                  <1>      ;
1361                  <1>      ;----- PUT CHARACTER INTO BUFFER
1362                  <1>      K57:      ; BUFFER_FILL
1363 000015EA 3CFF      <1>      cmp      al, -1      ; IS THIS AN IGNORE CHAR
1364 000015EC 7405      <1>      je      short K59      ; YES, DO NOTHING WITH IT
1365                  <1>      ;je     K26      ; YES, DO NOTHING WITH IT
1366 000015EE 80FCFF      <1>      cmp      ah, -1      ; LOOK FOR -1 PSEUDO SCAN
1367                  <1>      ;jne     short K61      ; NEAR_INTERRUPT_RETURN
1368                  <1>      ;je     K26      ; INTERRUPT_RETURN

```



```

1369      <1>      ; 12/04/2021
1370 000015F1 7505      <1>      jne short _K60      ; NEAR_INTERRUPT_RETURN
1371      <1>      K59:      <1>      ; NEAR_INTERRUPT_RETURN
1372 000015F3 E975FDFFFF      <1>      jmp      K26      ; INTERRUPT_RETURN
1373      <1>
1374      <1>      _K60: ; 29/01/2016
1375 000015F8 80FC68      <1>      cmp      ah, 68h ; ALT + F1 key
1376 000015FB 721D      <1>      jb      short K61
1377 000015FD 80FC6F      <1>      cmp      ah, 6Fh ; ALT + F8 key
1378 00001600 7718      <1>      ja      short K61
1379      <1>      ;
1380 00001602 8A1D[CE760100]      <1>      mov      bl, [ACTIVE_PAGE]
1381 00001608 80C368      <1>      add      bl, 68h
1382 0000160B 38E3      <1>      cmp      bl, ah
1383 0000160D 740B      <1>      je      short K61
1384      <1>      ; 24/07/2022
1385      <1>      ;push ax
1386 0000160F 50      <1>      push     eax
1387 00001610 88E0      <1>      mov      al, ah
1388 00001612 2C68      <1>      sub      al, 68h
1389 00001614 E896090000      <1>      call     set_active_page
1390 00001619 58      <1>      pop      eax
1391      <1>      ;pop ax
1392      <1>      K61:      <1>      ; NOT-CAPS-STATE
1393 0000161A 8B1D[FA670000]      <1>      mov      ebx, [BUFFER_TAIL]      ; GET THE END POINTER TO THE BUFFER
1394 00001620 89DE      <1>      mov      esi, ebx      ; SAVE THE VALUE
1395 00001622 E8B5FAFFFF      <1>      call     _K4      ; ADVANCE THE TAIL
1396 00001627 3B1D[F6670000]      <1>      cmp      ebx, [BUFFER_HEAD]      ; HAS THE BUFFER WRAPPED AROUND
1397 0000162D 740E      <1>      je      short K62      ; BUFFER_FULL_BEEP
1398 0000162F 668906      <1>      mov      [esi], ax      ; STORE THE VALUE
1399 00001632 891D[FA670000]      <1>      mov      [BUFFER_TAIL], ebx      ; MOVE THE POINTER UP
1400 00001638 E930FDFFFF      <1>      jmp      K26
1401      <1>      ;cli      ; TURN OFF INTERRUPTS
1402      <1>      ;mov      al, EOI      ; END OF INTERRUPT COMMAND
1403      <1>      ;out      INTA00, al      ; SEND COMMAND TO INTERRUPT CONTROL PORT
1404      <1>      ;mov      al, ENA_KBD      ; INSURE KEYBOARD IS ENABLED
1405      <1>      ;call     SHIP_IT      ; EXECUTE ENABLE
1406      <1>      ;mov      ax, 9102h      ; MOVE IN POST CODE & TYPE
1407      <1>      ;int      15h      ; PERFORM OTHER FUNCTION
1408      <1>      ;and      byte [KB_FLAG_3], ~(LC_E0+LC_E1) ; RESET LAST CHAR H.C. FLAG
1409      <1>      ;jmp      K27A      ; INTERRUPT_RETURN
1410      <1>      ;jmp      K27
1411      <1>      ;
1412      <1>      ;----- BUFFER IS FULL SOUND THE BEEPER
1413      <1>      K62:
1414 0000163D B020      <1>      mov      al, EOI      ; ENABLE INTERRUPT CONTROLLER CHIP
1415 0000163F E620      <1>      out      INTA00, al
1416 00001641 66B9A602      <1>      mov      cx, 678      ; DIVISOR FOR 1760 HZ
1417 00001645 B304      <1>      mov      bl, 4      ; SHORT BEEP COUNT (1/16 + 1/64 DELAY)
1418 00001647 E8880D0000      <1>      call     beep      ; GO TO COMMON BEEP HANDLER
1419 0000164C E928FDFFFF      <1>      jmp      K27      ; EXIT
1420      <1>
1421      <1>      SHIP_IT:
1422      <1>      ;-----
1423      <1>      ; SHIP_IT
1424      <1>      ; THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
1425      <1>      ; TO THE KEYBOARD CONTROLLER.
1426      <1>      ;-----
1427      <1>      ;
1428      <1>
1429      <1>      ;push ax      ; SAVE DATA TO SEND
1430      <1>      ; 12/04/2021
1431 00001651 50      <1>      push     eax
1432      <1>
1433      <1>      ;----- WAIT FOR COMMAND TO ACCEPTED
1434 00001652 FA      <1>      cli      ; DISABLE INTERRUPTS TILL DATA SENT
1435      <1>      ; xor      ecx, ecx      ; CLEAR TIMEOUT COUNTER
1436 00001653 B900000100      <1>      mov      ecx, 10000h
1437      <1>      S10:
1438 00001658 E464      <1>      in      al, STATUS_PORT      ; READ KEYBOARD CONTROLLER STATUS
1439 0000165A A802      <1>      test     al, INPT_BUF_FULL      ; CHECK FOR ITS INPUT BUFFER BUSY
1440 0000165C E0FA      <1>      loopnz   S10      ; WAIT FOR COMMAND TO BE ACCEPTED
1441      <1>
1442      <1>      ;pop ax      ; GET DATA TO SEND
1443      <1>      ; 12/04/2021
1444 0000165E 58      <1>      pop      eax
1445      <1>
1446 0000165F E664      <1>      out      STATUS_PORT, al      ; SEND TO KEYBOARD CONTROLLER
1447 00001661 FB      <1>      sti      ; ENABLE INTERRUPTS AGAIN
1448 00001662 C3      <1>      retn      ; RETURN TO CALLER
1449      <1>
1450      <1>      ; 12/04/2021 (32 bit push/pop)
1451      <1>      SND_DATA:
1452      <1>      ;-----
1453      <1>      ; SND_DATA
1454      <1>      ; THIS ROUTINES HANDLES TRANSMISSION OF COMMAND AND DATA BYTES
1455      <1>      ; TO THE KEYBOARD AND RECEIPT OF ACKNOWLEDGEMENTS. IT ALSO
1456      <1>      ; HANDLES ANY RETRIES IF REQUIRED
1457      <1>      ;-----
1458      <1>      ;
1459 00001663 50      <1>      push     eax ; push ax      ; SAVE REGISTERS
1460 00001664 53      <1>      push     ebx ; push bx
1461 00001665 51      <1>      push     ecx
1462 00001666 88C7      <1>      mov      bh, al      ; SAVE TRANSMITTED BYTE FOR RETRIES
1463 00001668 B303      <1>      mov      bl, 3      ; LOAD RETRY COUNT
1464      <1>      SD0:
1465 0000166A FA      <1>      cli      ; DISABLE INTERRUPTS
1466 0000166B 8025[EB670000]CF      <1>      and      byte [KB_FLAG_2], ~(KB_FE+KB_FA) ; CLEAR ACK AND RESEND FLAGS
1467      <1>      ;
1468      <1>      ;----- WAIT FOR COMMAND TO BE ACCEPTED
1469 00001672 B900000100      <1>      mov      ecx, 10000h      ; MAXIMUM WAIT COUNT
1470      <1>      SD5:
1471 00001677 E464      <1>      in      al, STATUS_PORT      ; READ KEYBOARD PROCESSOR STATUS PORT
1472 00001679 A802      <1>      test     al, INPT_BUF_FULL      ; CHECK FOR ANY PENDING COMMAND
1473 0000167B E0FA      <1>      loopnz   SD5      ; WAIT FOR COMMAND TO BE ACCEPTED
1474      <1>      ;
1475 0000167D 88F8      <1>      mov      al, bh      ; REESTABLISH BYTE TO TRANSMIT
1476 0000167F E660      <1>      out      PORT_A, al      ; SEND BYTE
1477 00001681 FB      <1>      sti      ; ENABLE INTERRUPTS
1478      <1>      ;mov      cx, 01A00h      ; LOAD COUNT FOR 10 ms+
1479 00001682 B9FFFF0000      <1>      mov      ecx, 0FFFFh
1480      <1>      SD1:
1481 00001687 F605[EB670000]30      <1>      test     byte [KB_FLAG_2], KB_FE+KB_FA ; SEE IF EITHER BIT SET
1482 0000168E 750F      <1>      jnz      short SD3      ; IF SET, SOMETHING RECEIVED GO PROCESS
1483 00001690 E2F5      <1>      loop     SD1      ; OTHERWISE WAIT
1484      <1>      SD2:
1485 00001692 FECB      <1>      dec      bl      ; DECREMENT RETRY COUNT
1486 00001694 75D4      <1>      jnz      short SD0      ; RETRY TRANSMISSION
1487 00001696 800D[EB670000]80      <1>      or      byte [KB_FLAG_2], KB_ERR ; TURN ON TRANSMIT ERROR FLAG
1488 0000169D EB09      <1>      jmp      short SD4      ; RETRIES EXHAUSTED FORGET TRANSMISSION
1489      <1>      SD3:
1490 0000169F F605[EB670000]10      <1>      test     byte [KB_FLAG_2], KB_FA ; SEE IF THIS IS AN ACKNOWLEDGE
1491 000016A6 74EA      <1>      jz      short SD2      ; IF NOT, GO RESEND
1492      <1>      SD4:

```

```

1493 000016A8 59      <1>      pop     ecx                ; RESTORE REGISTERS
1494 000016A9 5B      <1>      pop     ebx ; pop bx
1495 000016AA 58      <1>      pop     eax ; pop ax
1496 000016AB C3      <1>      retn                ; RETURN, GOOD TRANSMISSION
1497
1498
1499
1500
1501
1502
1503
1504
1505 000016AC FA      <1>      cli                ; TURN OFF INTERRUPTS
1506 000016AD F605[EB670000]40 <1>      test    byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
1507 000016B4 755F      <1>      jnz     short SL1        ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
1508
1509 000016B6 800D[EB670000]40 <1>      or      byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
1510 000016BD B020      <1>      mov     al, EOI          ; END OF INTERRUPT COMMAND
1511 000016BF E620      <1>      out     20h, al ;out INTA00, al ; SEND COMMAND TO INTERRUPT CONTROL PORT
1512 000016C1 EB11      <1>      jmp     short SL0        ; GO SEND MODE INDICATOR COMMAND
1513
1514 000016C3 FA      <1>      cli                ; TURN OFF INTERRUPTS
1515 000016C4 F605[EB670000]40 <1>      test    byte [KB_FLAG_2], KB_PR_LED ; CHECK FOR MODE INDICATOR UPDATE
1516 000016CB 7548      <1>      jnz     short SL1        ; DON'T UPDATE AGAIN IF UPDATE UNDERWAY
1517
1518 000016CD 800D[EB670000]40 <1>      or      byte [KB_FLAG_2], KB_PR_LED ; TURN ON UPDATE IN PROCESS
1519
1520 000016D4 B0ED      <1>      mov     al, LED_CMD        ; LED CMD BYTE
1521 000016D6 E888FFFFFF      <1>      call    SND_DATA        ; SEND DATA TO KEYBOARD
1522 000016DB FA      <1>      cli                ; TURN OFF INTERRUPTS
1523 000016DC E836000000      <1>      call    MAKE_LED        ; GO FORM INDICATOR DATA BYTE
1524 000016E1 8025[EB670000]F8 <1>      and     byte [KB_FLAG_2], 0F8h ; ~KB_LEDS ; CLEAR MODE INDICATOR BITS
1525 000016E8 0805[EB670000]      <1>      or      [KB_FLAG_2], al      ; SAVE PRESENT INDICATORS FOR NEXT TIME
1526 000016EE F605[EB670000]80 <1>      test    byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
1527 000016F5 750F      <1>      jnz     short SL2        ; IF SO, BYPASS SECOND BYTE TRANSMISSION
1528
1529 000016F7 E867FFFFFF      <1>      call    SND_DATA        ; SEND DATA TO KEYBOARD
1530 000016FC FA      <1>      cli                ; TURN OFF INTERRUPTS
1531 000016FD F605[EB670000]80 <1>      test    byte [KB_FLAG_2], KB_ERR ; TRANSMIT ERROR DETECTED
1532 00001704 7408      <1>      jz      short SL3        ; IF NOT, DON'T SEND AN ENABLE COMMAND
1533
1534 00001706 B0F4      <1>      mov     al, KB_ENABLE      ; GET KEYBOARD CSA ENABLE COMMAND
1535 00001708 E856FFFFFF      <1>      call    SND_DATA        ; SEND DATA TO KEYBOARD
1536 0000170D FA      <1>      cli                ; TURN OFF INTERRUPTS
1537
1538 0000170E 8025[EB670000]3F <1>      and     byte [KB_FLAG_2], ~(KB_PR_LED+KB_ERR) ; TURN OFF MODE INDICATOR
1539
1540 00001715 FB      <1>      sti                ; UPDATE AND TRANSMIT ERROR FLAG
1541 00001716 C3      <1>      retn                ; ENABLE INTERRUPTS
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551 00001717 A0[E9670000] <1>      push    cx                ; SAVE CX
1552 0000171C 2470      <1>      mov     al, [KB_FLAG]      ; GET CAPS & NUM LOCK INDICATORS
1553
1554
1555 0000171E C0C004      <1>      and     al, CAPS_STATE+NUM_STATE+SCROLL_STATE ; ISOLATE INDICATORS
1556 00001721 2407      <1>      ;mov     cl, 4                ; SHIFT COUNT
1557
1558 00001723 C3      <1>      ;rol     al, cl                ; SHIFT BITS OVER TO TURN ON INDICATORS
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
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
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
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
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3
```

```

52      (AL) = 06H 640x200 BW MODE :
53      (AL) = 07H 80x25 MONOCHROME (USED INTERNAL TO VIDEO ONLY) :
54      *** NOTES -BW MODES OPERATE SAME AS COLOR MODES, BUT COLOR :
55      BURST IS NOT ENABLED :
56      -CURSOR IS NOT DISPLAYED IN GRAPHICS MODE :
57      (AH)= 01H SET CURSOR TYPE :
58      (CH) = BITS 4-0 = START LINE FOR CURSOR :
59      ** HARDWARE WILL ALWAYS CAUSE BLINK :
60      ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING :
61      OR NO CURSOR AT ALL :
62      (CL) = BITS 4-0 = END LINE FOR CURSOR :
63      (AH)= 02H SET CURSOR POSITION :
64      (DH,DL) = ROW,COLUMN (00H,00H) IS UPPER LEFT :
65      (BH) = A PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES) :
66      (AH)= 03H READ CURSOR POSITION :
67      (BH) = PAGE NUMBER (MUST BE 00H FOR GRAPHICS MODES) :
68      ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR :
69      (CH,CL) = CURSOR MODE CURRENTLY SET :
70      (AH)= 04H READ LIGHT PEN POSITION :
71      ON EXIT: :
72      (AH) = 00H -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED :
73      (AH) = 01H -- VALID LIGHT PEN VALUE IN REGISTERS :
74      (DH,DL) = ROW,COLUMN OF CHARACTER LP POSITION :
75      (CH) = RASTER LINE (0-199) :
76      (BX) = PIXEL COLUMN (0-319,639) :
77      (AH)= 05H SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR ALPHA MODES) :
78      (AL) = NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR MODES 2&3) :
79      (AH)= 06H SCROLL ACTIVE PAGE UP :
80      (AL) = NUMBER OF LINES. ( LINES BLANKED AT BOTTOM OF WINDOW ) :
81      (AL) = 00H MEANS BLANK ENTIRE WINDOW :
82      (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL :
83      (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL :
84      (BH) = ATTRIBUTE TO BE USED ON BLANK LINE :
85      (AH)= 07H SCROLL ACTIVE PAGE DOWN :
86      (AL) = NUMBER OF LINES, INPUT LINES BLANKED AT TOP OF WINDOW :
87      (AL) = 00H MEANS BLANK ENTIRE WINDOW :
88      (CH,CL) = ROW,COLUMN OF UPPER LEFT CORNER OF SCROLL :
89      (DH,DL) = ROW,COLUMN OF LOWER RIGHT CORNER OF SCROLL :
90      (BH) = ATTRIBUTE TO BE USED ON BLANK LINE :
91      :
92      CHARACTER HANDLING ROUTINES :
93      :
94      (AH)= 08H READ ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION :
95      (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) :
96      ON EXIT: :
97      (AL) = CHAR READ :
98      (AH) = ATTRIBUTE OF CHARACTER READ (ALPHA MODES ONLY) :
99      (AH)= 09H WRITE ATTRIBUTE/CHARACTER AT CURRENT CURSOR POSITION :
100      (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) :
101      (CX) = COUNT OF CHARACTERS TO WRITE :
102      (AL) = CHAR TO WRITE :
103      (BL) = ATTRIBUTE OF CHARACTER (ALPHA)/COLOR OF CHAR (GRAPHICS) :
104      SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1. :
105      (AH) = 0AH WRITE CHARACTER ONLY AT CURRENT CURSOR POSITION :
106      (BH) = DISPLAY PAGE (VALID FOR ALPHA MODES ONLY) :
107      (CX) = COUNT OF CHARACTERS TO WRITE :
108      (AL) = CHAR TO WRITE :
109      NOTE: USE FUNCTION (AH)= 09H IN GRAPHICS MODES :
110      FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE, THE :
111      CHARACTERS ARE FORMED FROM A CHARACTER GENERATOR IMAGE :
112      MAINTAINED IN THE SYSTEM ROM. ONLY THE 1ST 128 CHARS :
113      ARE CONTAINED THERE. TO READ/WRITE THE SECOND 128 CHARS, :
114      THE USER MUST INITIALIZE THE POINTER AT INTERRUPT 1FH, :
115      (LOCATION 0007CH) TO POINT TO THE 1K BYTE TABLE CONTAINING :
116      THE CODE POINTS FOR THE SECOND 128 CHARS (128-255). :
117      FOR WRITE CHARACTER INTERFACE IN GRAPHICS MODE, THE REPLICATION FACTOR :
118      CONTAINED IN (CX) ON ENTRY WILL PRODUCE VALID RESULTS ONLY :
119      FOR CHARACTERS CONTAINED ON THE SAME ROW. CONTINUATION TO :
120      SUCCEEDING LINES WILL NOT PRODUCE CORRECTLY. :
121      :
122      GRAPHICS INTERFACE :
123      (AH)= 0BH SET COLOR PALETTE :
124      (BH) = PALETTE COLOR ID BEING SET (0-127) :
125      (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID :
126      NOTE: FOR THE CURRENT COLOR CARD, THIS ENTRY POINT HAS :
127      MEANING ONLY FOR 320x200 GRAPHICS. :
128      COLOR ID = 0 SELECTS THE BACKGROUND COLOR (0-15) :
129      COLOR ID = 1 SELECTS THE PALETTE TO BE USED: :
130      0 = GREEN(1)/RED(2)/YELLOW(3) :
131      1 = CYAN(1)/MAGENTA(2)/WHITE(3) :
132      IN 40x25 OR 80x25 ALPHA MODES, THE VALUE SET FOR :
133      PALETTE COLOR 0 INDICATES THE BORDER COLOR :
134      TO BE USED (VALUES 0-31, WHERE 16-31 SELECT :
135      THE HIGH INTENSITY BACKGROUND SET. :
136      (AH)= 0CH WRITE DOT :
137      (DX) = ROW NUMBER :
138      (CX) = COLUMN NUMBER :
139      (AL) = COLOR VALUE :
140      IF BIT 7 OF AL = 1, THEN THE COLOR VALUE IS EXCLUSIVE :
141      Ored WITH THE CURRENT CONTENTS OF THE DOT :
142      (AH)= 0DH READ DOT :
143      (DX) = ROW NUMBER :
144      (CX) = COLUMN NUMBER :
145      (AL) = RETURNS THE DOT READ :
146      :
147      ASCII TELETYPE ROUTINE FOR OUTPUT :
148      :
149      (AH)= 0EH WRITE TELETYPE TO ACTIVE PAGE :
150      (AL) = CHAR TO WRITE :
151      (BL) = FOREGROUND COLOR IN GRAPHICS MODE :
152      NOTE -- SCREEN WIDTH IS CONTROLLED BY PREVIOUS MODE SET :
153      (AH)= 0FH CURRENT VIDEO STATE :
154      RETURNS THE CURRENT VIDEO STATE :
155      (AL) = MODE CURRENTLY SET ( SEE (AH)=00H FOR EXPLANATION) :
156      (AH) = NUMBER OR CHARACTER COLUMNS ON SCREEN :
157      (BH) = CURRENT ACTIVE DISPLAY PAGE :
158      (AH)= 10H RESERVED :
159      (AH)= 11H RESERVED :
160      (AH)= 12H RESERVED :
161      (AH)= 13H WRITE STRING :
162      ES:BP - POINTER TO STRING TO BE WRITTEN :
163      CX - LENGTH OF CHARACTER STRING TO WRITTEN :
164      DX - CURSOR POSITION FOR STRING TO BE WRITTEN :
165      BH - PAGE NUMBER :
166      (AL)= 00H WRITE CHARACTER STRING :
167      BL - ATTRIBUTE :
168      STRING IS <CHAR,CHAR, ... ,CHAR> :
169      CURSOR NOT MOVED :
170      (AL)= 01H WRITE CHARACTER STRING AND MOVE CURSOR :
171      BL - ATTRIBUTE :
172      STRING IS <CHAR,CHAR, ... ,CHAR> :
173      CURSOR MOVED :
174      (AL)= 02H WRITE CHARACTER AND ATTRIBUTE STRING :
175      (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      0000172C [431B0000] <1> M1: dd      SET_MODE          ; TABLE OF ROUTINES WITHIN VIDEO I/O
191      00001730 [F91E0000] <1>          dd      SET_CTYPE
192      00001734 [271F0000] <1>          dd      SET_CPOS
193      00001738 [4B1F0000] <1>          dd      READ_CURSOR
194      <1>          ;dd      VIDEO_RETURN      ; READ_LPEN
195      0000173C [411B0000] <1>          dd      set_mode_ncm      ; Set mode without clearing video memory
196      00001740 [911F0000] <1>          dd      ACT_DISP_PAGE
197      00001744 [1B200000] <1>          dd      SCROLL_UP
198      00001748 [32210000] <1>          dd      SCROLL_DOWN
199      0000174C [B1210000] <1>          dd      READ_AC_CURRENT
200      00001750 [0E220000] <1>          dd      WRITE_AC_CURRENT
201      00001754 [34220000] <1>          dd      WRITE_C_CURRENT
202      00001758 [3C2B0000] <1>          dd      SET_COLOR
203      0000175C [A42B0000] <1>          dd      WRITE_DOT
204      00001760 [732B0000] <1>          dd      READ_DOT
205      00001764 [BE220000] <1>          dd      WRITE_TTY
206      00001768 [291B0000] <1>          dd      VIDEO_STATE
207      0000176C [71360000] <1>          dd      vga_pal_funcs      ; 10/08/2016 (TRDOS 386)
208      00001770 [89300000] <1>          dd      font_setup        ; 10/07/2016 (TRDOS 386)
209      00001774 [781B0000] <1>          dd      VIDEO_RETURN      ; RESERVED
210      00001778 [34240000] <1>          dd      WRITE_STRING      ; 23/06/2016 (TRDOS 386)
211      <1>          M1L EQU      $ - M1
212      <1> ;
213      <1> ; 22/12/2025
214      <1> ; 21/12/2025 - TRDOS 386 v2.0.10
215      <1> ; 29/11/2023 - TRDOS 386 v2.0.7
216      <1> ; [VESA VBE3-PMI functions]
217      <1> ; (fixing page table problems for LFB/PMI for >2.5GB main memory)
218      <1> ; ((Addresses of kernel page tables and PMI videobios address
219      <1> ; must be < 4MB or cr3 register must contain kernel's page dir
220      <1> ; during video interrupt. Save-change-restore cr3 register
221      <1> ; content here is a bugfix method to prevent page faults
222      <1> ; for -real- computers with >2.5GB main memory.))
223      <1> ; {INT 31h was not changing cr3 while it contains user's
224      <1> ; page directory addr and accessing beyond of the 1st 4MB was
225      <1> ; causing to page faults.. because, after the 1st 4MB, user's
226      <1> ; page tables contain non-linear/virtual memory pages.}
227      <1> ;
228      <1> ; 02/08/2022 - TRDOS 386 v2.0.5
229      <1> ; 06/12/2020
230      <1> ; 05/12/2020
231      <1> ; 03/12/2020
232      <1> ; 27/11/2020 - TRDOS 386 v2.0.3
233      <1> ; 14/01/2017
234      <1> ; 02/01/2017
235      <1> ; 04/07/2016
236      <1> ; 12/05/2016
237      <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
238      <1> int31h: ; Video BIOS
239      <1> ;
240      <1> ; BH = Video page number
241      <1> ; BL = Color/Attribute
242      <1> ; AH = Function number
243      <1> ; AL = Character
244      <1> ;
245      <1> VIDEO_IO_1:
246      <1> ;sti                                ; INTERRUPTS BACK ON
247      0000177C FC <1> cld                                ; SET DIRECTION FORWARD
248      <1> ;
249      <1> ;cmp      ah, M1L/4                    ; TEST FOR WITHIN TABLE RANGE
250      <1> ;jnb      short M4                    ; BRANCH TO EXIT IF NOT A VALID COMMAND
251      <1> ;
252      <1> ; 26/11/2020
253      0000177D 80FC14 <1> cmp      ah, M1L/4
254      00001780 7205 <1> jb       short VGA_func
255      <1> ;
256      00001782 80FC4F <1> cmp      ah, 4Fh
257      00001785 7536 <1> jne      short M4 ; invalid !
258      <1> ;
259      <1> VGA_func: ; 26/11/2020
260      00001787 06 <1> push     es ; *
261      00001788 1E <1> push     ds ; **
262      <1> ;
263      <1> ; 26/11/2020
264      00001789 50 <1> push     eax ; -
265      <1> ;
266      0000178A 66B81000 <1> mov      ax, KDATA
267      0000178E 8ED8 <1> mov      ds, ax
268      00001790 8EC0 <1> mov      es, ax
269      <1> ;
270      <1> ; 26/11/2020
271      00001792 58 <1> pop      eax ; +
272      <1> ;
273      00001793 FB <1> sti
274      00001794 80FC4F <1> cmp      ah, 4Fh
275      <1> ;je      short VBE_func
276      <1> ; 21/12/2025
277      00001797 0F84BC000000 <1> je       VBE_func
278      <1> ;
279      <1> ; 04/12/2020
280      0000179D A3[28830100] <1> mov      [video_eax], eax
281      <1> ;
282      <1> ; 21/12/2020
283      000017A2 803D[1E680000]FF <1> cmp      byte [CRT_MODE], 0FFh ; Current mode is a VESA VBE mode ?
284      000017A9 7213 <1> jb       short VGA_func_std
285      <1> ;
286      000017AB 08E4 <1> or       ah, ah
287      000017AD 750F <1> jnz      short VGA_func_std ; set mode ?
288      <1> ;no
289      000017AF 803D[86090000]03 <1> cmp      byte [vbe3], 3
290      000017B6 7506 <1> jne      short VGA_func_std ; (real) VESA VBE 3 video bios ?
291      <1> ;no
292      000017B8 E9D7010000 <1> jmp      vesa_vbe3_pmi
293      <1> ;
294      <1> ; 21/12/2020
295      <1> M4: ; COMMAND NOT VALID
296      000017BD CF <1> iretd ; DO NOTHING IF NOT IN VALID RANGE
297      <1> ;
298      <1> VGA_func_std:
299      <1> ; 06/12/2020

```

```

300      <1>      ; 03/12/2020
301 000017BE 80FC0F <1>      cmp     ah, 0Fh
302      <1>      ;ja      short VGA_funcs_0      ; only CGA funcs will be handled by
303      <1>      ; 21/12/2025
304 000017C1 777F <1>      ja      VGA_funcs_0      ; vga bios firmware
305      <1>
306      <1>      ; 06/12/2020
307      <1>      ; 03/12/2020
308      <1>      ;test    ah, 7Fh ; set mode ?
309      <1>      ;;or     ah, ah      ; only 'set mode' will be handled by
310      <1>      ;jnz     short VGA_funcs_0      ; vga bios firmware
311      <1>      ;jz      short vbe_pmi32_0
312      <1>
313      <1>      ; 28/11/2020
314 000017C3 803D[0C9D0100]00 <1>      cmp     byte [pmi32], 0      ; 32 bit protected mode interface for
315 000017CA 7676 <1>      jna     short VGA_funcs_0      ; video hardware's vga bios firmware
316      <1>      ; ([pmi32] > 0 if it is activated)
317      <1>      ; note:
318      <1>      ; [vbe3] = 3 is required to activate
319      <1>      ; 07/12/2020
320 000017CC 20E4 <1>      and     ah, ah ; is this set mode ?
321 000017CE 7413 <1>      jz      short vbe_pmi32_2 ; yes
322      <1>
323 000017D0 80FC04 <1>      cmp     ah, 04h ; set mode ('no clear memory' option)
324 000017D3 746D <1>      je      short VGA_funcs_0
325      <1>
326      <1>      ; 07/12/2020
327 000017D5 803D[1E680000]07 <1>      cmp     byte [CRT_MODE], 7 ; current mode > 7 ?
328 000017DC 7664 <1>      jna     short VGA_funcs_0 ; no
329      <1>
330      <1>      ; when mode 3 is active,
331      <1>      ; video bios functions are not redirected
332      <1>      ; to VESA VBE3 PMI except 'set mode' function
333      <1>
334      <1>      vbe_pmi32_0:
335      <1>      ; 06/12/2020
336      <1>      ;or      ah, ah
337      <1>      ;jnz     short vbe_pmi32_2
338      <1>      ; ah = 0 ; 'set mode'
339      <1>      ;cmp     al, 3 ; 80x25 text mode, 16 colors, default mode for MainProg
340      <1>      ;jne     short vbe_pmi32_1
341      <1>      ;cmp     byte [CRT_MODE], 3 ; If current video mode <= 3 and requested
342      <1>      ; ; ; video mode is 3, use internal 'set mode';
343      <1>      ; ; ; otherwise, use vesa vbe 3 bios's 'set mode'.
344      <1>      ;jne     short VGA_funcs_0
345      <1>      vbe_pmi32_1:
346 000017DE E9B1010000 <1>      jmp     vesa_vbe3_pmi
347      <1>
348      <1>      ;vbe_pmi32_2:
349      <1>      ;cmp     ah, 04h ; set mode (no clear mem option)
350      <1>      ;jne     short vbe_pmi32_1
351      <1>
352      <1>      vbe_pmi32_2:
353      <1>
354      <1>      ; 22/12/2025
355      <1>      ; 21/12/2025 - TRDOS 386 v2.0.10
356      <1>      ; set character height (needed) for next system calls
357      <1>      ;;;;
358      <1>      vbe_pmi32_3:
359 000017E3 56 <1>      push    esi
360 000017E4 50 <1>      push    eax
361 000017E5 88C4 <1>      mov     ah, al
362 000017E7 80E47F <1>      and     ah, 7Fh
363 000017EA BE[3A680000] <1>      mov     esi, vga_modes
364      <1>      vbe_pmi32_3_1:
365 000017EF AC <1>      lodsb
366 000017F0 38C4 <1>      cmp     ah, al
367 000017F2 7408 <1>      je      short vbe_pmi32_3_2
368 000017F4 81FE[4A680000] <1>      cmp     esi, vga_modes+vga_mode_count
369 000017FA 72F3 <1>      jb      short vbe_pmi32_3_1
370      <1>      vbe_pmi32_3_2:
371 000017FC 81EE[3A680000] <1>      sub     esi, vga_modes
372 00001802 C1E602 <1>      shl     esi, 2 ; dword
373 00001805 81C6[4A680000] <1>      add     esi, vga_mode_tbl_ptr
374 00001808 8B36 <1>      mov     esi, [esi]
375 0000180D 66AD <1>      lodsw
376 0000180F A2[20680000] <1>      mov     [CRT_COLS], al
377 00001814 FEC4 <1>      inc     ah
378 00001816 8825[26680000] <1>      mov     [VGA_ROWS], ah
379 0000181C 8A06 <1>      mov     al, [esi]
380      <1>      ;lodsb
381 0000181E A2[22680000] <1>      mov     [CHAR_HEIGHT], al
382      <1>      vbe_pmi32_4:
383 00001823 58 <1>      pop     eax
384 00001824 5E <1>      pop     esi
385      <1>      ;;;;
386      <1>
387      <1>      ; 07/12/2020
388 00001825 803D[1E680000]07 <1>      cmp     byte [CRT_MODE], 7 ; current mode > 7 ?
389 0000182C 77B0 <1>      ja      short vbe_pmi32_1 ; yes
390      <1>
391 0000182E 3C07 <1>      cmp     al, 7 ; requested mode > 7 ?
392 00001830 7610 <1>      jna     short VGA_funcs_0 ; no (CGA)
393      <1>
394 00001832 3C13 <1>      cmp     al, 13h
395 00001834 76A8 <1>      jna     short vbe_pmi32_1
396      <1>
397 00001836 A880 <1>      test    al, 80h
398 00001838 7408 <1>      jz      short VGA_funcs_0 ; unknown or special
399      <1>
400 0000183A 3C87 <1>      cmp     al, 87h ; requested mode > 7 ?
401      <1>      ; (with no clear mem ops)
402 0000183C 7604 <1>      jna     short VGA_funcs_0 ; no (CGA)
403      <1>
404 0000183E 3C93 <1>      cmp     al, 93h
405 00001840 769C <1>      jna     short vbe_pmi32_1
406      <1>
407      <1>      ; > 13h video modes are unknown or special
408      <1>      ; they must be handled by kernel
409      <1>
410      <1>      ; CGA video modes will be handled by kernel
411      <1>
412      <1>      VGA_funcs_0:
413 00001842 52 <1>      push    edx ; ***
414 00001843 51 <1>      push    ecx ; ****
415 00001844 53 <1>      push    ebx ; *****
416 00001845 56 <1>      push    esi ; *****
417 00001846 57 <1>      push    edi ; *****
418 00001847 55 <1>      push    ebp ; *****
419      <1>
420      <1>      ;mov     [video_eax], eax ; 12/05/2016
421 00001848 BF00800B00 <1>      mov     edi, 0B8000h ; GET offset FOR COLOR CARD
422      <1>
423      <1>      ; 23/03/2016

```

```

424 0000184D C0E402    <1>    shl     ah, 2    ; dword      ; TIMES 2 FOR WORD TABLE LOOKUP
425 00001850 0FB6F4    <1>    movzx   esi, ah    ; MOVE OFFSET INTO LOOK UP REGISTER (SI)
426                                <1>    ;mov     ah, [CRT_MODE]    ; MOVE CURRENT MODE INTO (AH) REGISTER
427                                <1>
428                                <1>    ; 15/01/2017
429                                <1>    ; 14/01/2017
430                                <1>    ; 02/01/2017
431                                <1>    ; mov     byte [intflg], 31h ; video interrupt
432                                <1>    ; sti     ; 26/11/2020
433                                <1>    ;
434                                <1>
435 00001853 FFA6[2C170000] <1>    jmp     dword [esi+M1]    ; GO TO SELECTED FUNCTION
436                                <1>
437                                <1> VBE_func:
438                                <1>    ; 26/11/2020
439                                <1>    ; sti
440 00001859 55          <1>    push    ebp ; *** ; 27/11/2020
441 0000185A 56          <1>    push    esi ; ****
442                                <1>
443                                <1>    ; Note:
444                                <1>    ; ebx, ecx, edx, edi, ebp registers
445                                <1>    ; must be saved by VBE functions and
446                                <1>    ; eax register must be set
447                                <1>    ; (according to VBE 3 standard)
448                                <1>    ; before return from this interrupt
449                                <1>    ; (every function must restore and set
450                                <1>    ; registers except esp, esi, es, ds)
451                                <1>
452                                <1>    ; 29/11/2023 - TRDOS 386 v2.0.7
453 0000185B 0F20DE    <1>    mov     esi, cr3
454 0000185E 56          <1>    push    esi ; *****
455                                <1>    ; cmp     esi, [k_page_dir]
456                                <1>    ; je      short VBE_func_x
457 0000185F 8B35[A0760100] <1>    mov     esi, [k_page_dir]
458 00001865 0F22DE    <1>    mov     cr3, esi
459                                <1> ;VBE_func_x:
460 00001868 803D[86090000]02 <1>    cmp     byte [vbe3], 2
461 0000186F 7744          <1>    ja      short VESA_VBE3_PMI_CALL ; VBE3 video bios ('PMID')
462                                <1>    ; je      short VBE_func_0 ; Bochs/Qemu/VirtualBox emulator
463 00001871 726C          <1>    jb      short VBE_unknown ; invalid ([vbe3] = 0)
464                                <1>
465                                <1>    ; jmp     VESA_VBE3_PMI_CALL
466                                <1>
467                                <1> VBE_func_0:
468                                <1>    ; Bochs/Plex86 VGAbios VBE extension
469                                <1>    ; (TRDOS 386 v2.0.3 can use VBE graphics modes on emulators)
470                                <1>    ; BOCHS/QEMU/VIRTUALBOX
471                                <1>
472 00001873 8A25[87090000] <1>    mov     ah, [vbe2bios]
473 00001879 80FCC0    <1>    cmp     ah, 0C0h
474 0000187C 7261          <1>    jb      short VBE_unknown
475 0000187E 80FCC5    <1>    cmp     ah, 0C5h
476 00001881 775C          <1>    ja      short VBE_unknown
477                                <1>
478                                <1>    ; TRDOS 386 is running on BOCHS or QEMU
479 00001883 B44F          <1>    mov     ah, 4Fh
480                                <1> VBE_func_1:
481 00001885 0FB6F0    <1>    movzx   esi, al ; VESA VBE function number
482                                <1>    ; shl     si, 2 ; dword
483                                <1>    ; 02/08/2022
484 00001888 C1E602    <1>    shl     esi, 2
485 0000188B 6683FE14    <1>    cmp     si, NIL
486 0000188F 734E          <1>    jnb     short VBE_unknown
487                                <1>    ; sti
488                                <1>
489 00001891 FF96[A1180000] <1>    call    dword [esi+N1] ; call VBE function
490                                <1>
491                                <1>    ; jmp     short VBE_bios_return
492                                <1>
493                                <1> VBE_bios_return:
494 00001897 FA          <1>    cli
495                                <1>    ; 29/11/2023 - TRDOS 386 v2.0.7
496 00001898 5E          <1>    pop     esi ; *****
497                                <1>    ; cmp     esi, [k_page_dir]
498                                <1>    ; je      short VBE_bios_return_x
499 00001899 0F22DE    <1>    mov     cr3, esi
500                                <1> ;VBE_bios_return_x:
501 0000189C 5E          <1>    pop     esi ; ****
502 0000189D 5D          <1>    pop     ebp ; *** ; 27/11/2020
503 0000189E 07          <1>    pop     es ; **
504 0000189F 1F          <1>    pop     ds ; *
505 000018A0 CF          <1>    iretd
506                                <1>
507                                <1>    ; 26/11/2020
508                                <1>
509 000018A1 [02190000] <1>    N1:    dd     vbe_biosfn_return_ctrl_info
510 000018A5 [29390000] <1>    dd     vbe_biosfn_return_mode_info
511 000018A9 [E3390000] <1>    dd     vbe_biosfn_set_mode
512 000018AD [B33A0000] <1>    dd     vbe_biosfn_return_current_mode
513 000018B1 [D23A0000] <1>    dd     vbe_biosfn_save_restore_state
514                                <1>    ; dd     vbe_biosfn_display_window_ctrl
515                                <1>    ; dd     vbe_biosfn_set_get_log_scanline
516                                <1>    ; dd     vbe_biosfn_set_get_disp_start
517                                <1>    ; dd     vbe_biosfn_set_get_dac_pal_frm
518                                <1>    ; dd     vbe_biosfn_set_get_palette_data
519                                <1>
520                                <1> N1L EQU    $ - N1
521                                <1>
522                                <1> VESA_VBE3_PMI_CALL: ; VESA VBE video bios (firmware) functions
523                                <1>    ; by using VESA VBE3 Protected Mode Interface
524                                <1>
525                                <1>    ; 02/08/2022 - TRDOS 386 v2.0.5
526                                <1>    ; 29/11/2020
527                                <1>    ; 26/11/2020 - TRDOS 386 v2.0.3 video.s
528                                <1>
529                                <1>    ; We are here because..
530                                <1>    ; 'PMID' has been verified by TRDOS 386 v2.0.3 kernel.
531                                <1>    ; (Otherwise bochs/plex86 compatible VBE functions and
532                                <1>    ; modes would be used on BOCHS/QEMU/VIRTUALBOX emulators
533                                <1>    ; or only standard/old VGA graphics modes would be used.)
534                                <1>
535                                <1>    ; (TRDOS 386 v2.0.3 can use VESA VBE graphics modes if
536                                <1>    ; the video bios is full compatible with VBE3 standard)
537                                <1>
538                                <1>    ; 29/11/2020
539                                <1>
540 000018B5 0FB6F0    <1>    movzx   esi, al ; VESA VBE 3 function number
541                                <1>    ; shl     si, 2 ; dword
542                                <1>    ; 02/08/2022
543 000018B8 C1E602    <1>    shl     esi, 2
544 000018BB 6683FE14    <1>    cmp     si, NIL
545 000018BF 731E          <1>    jnb     short VBE_unknown
546                                <1>    ; sti
547                                <1>

```

```

548 000018C1 57      <1>      push    edi ; *****
549                <1>
550 000018C2 FF96[CB180000] <1>      call    dword [esi+P1] ; call VBE 3 function
551                <1>
552 000018C8 5F      <1>      pop     edi ; *****
553                <1>
554 000018C9 EBCC     <1>      jmp     short VBE_bios_return
555                <1>
556                <1> P1:
557 000018CB [E5180000] <1>      dd      vbe3_pmf_n_return_ctrl_info
558 000018CF [08190000] <1>      dd      vbe3_pmf_n_return_mode_info
559 000018D3 [44190000] <1>      dd      vbe3_pmf_n_set_mode
560 000018D7 [3F190000] <1>      dd      vbe3_pmf_n_return_current_mode
561 000018DB [4D1A0000] <1>      dd      vbe3_pmf_n_save_restore_state
562                <1> ;dd      vbe3_pmf_n_display_window_ctrl
563                <1> ;dd      vbe3_pmf_n_set_get_log_scanline
564                <1> ;dd      vbe3_pmf_n_set_get_disp_start
565                <1> ;dd      vbe3_pmf_n_set_get_dac_pal_frm
566                <1> ;dd      vbe3_pmf_n_set_get_palette_data
567                <1> ;dd      vbe3_pmf_n_return_pmi ; invalid for TRDOS 386 v2
568                <1> ;dd      vbe3_pmf_n_set_get_pixel_clock
569                <1>
570                <1> P1L EQU    $ - P1
571                <1>
572                <1> ; 29/11/2020
573                <1> ; mov     edi, VBE3MODEINFOBLOCK >> 4 ; / 16
574                <1> ;
575                <1> ; cmp     al, 04h
576                <1> ; jb      short vbe3_pm_f ; function: 4F00h to 4F03h
577                <1> ; ja      short vbe3_pmi_f5B ; function: 4F05h to 4F0Bh
578                <1> ;
579                <1> ; check buffer length (must be <= 2048 bytes)
580                <1> ;
581                <1> ; and     dl, dl ; 0
582                <1> ; jz      short vbe3_pm_f
583                <1> ; ; Return Save/Restore State buffer size
584                <1> ;
585                <1> ; push    ebx ; buffer address
586                <1> ; push    edx ; function: save (01h) or restore (02h)
587                <1> ; call
588                <1> ;
589                <1> ; vbe3_pm_f03:
590                <1> ; cmp     al, 2
591                <1> ; ja      short vbe3_pm_f ; function 4F03h
592                <1> ; jb      short vbe3_pm_f1
593                <1> ;
594                <1> ;
595                <1> ; vbe3_pm_f1:
596                <1> ;
597                <1> ;
598                <1> ; vbe3_pmi_f5B:
599                <1> ; cmp     al, 09h
600                <1> ; jna     short vbe3_pm_f ; funcs 05h to 09h are usable
601                <1> ;
602                <1> ; cmp     al, 0Bh ; Get/Set pixel clock, last function
603                <1> ; jne     short VBE_unknown
604                <1> ; ; (do not use 'uncertain' functions
605                <1> ; ; because of system-user buff transfers)
606                <1> ; vbe3_pm_f:
607                <1> ; mov     byte [vbe3_pm_fn], al ; set
608                <1> ; ; prepare 16 bit pm segments & registers for pmi call
609                <1> ; call    VESA_VBE3_PM_FUNCTION
610                <1> ;
611                <1> ;
612                <1> ;
613                <1> ; 26/11/2020
614                <1> ; VBE_unknown:
615 000018DF 66B80001 <1>      mov     ax, 0100h ; ah = 1 : Function call failed
616                <1> ; ; al = 0 : Function is not supported
617                <1> ;
618 000018E3 EBB2     <1>      jmp     short VBE_bios_return
619                <1>
620                <1> vbe3_pmf_n_return_ctrl_info:
621                <1> ; 02/08/2022 (TRDOS 386 kernel v2.0.5)
622                <1> ; 12/12/2020
623                <1> ;
624                <1> ; VBE function 4F00h - Return VBE Controller Information
625                <1> ;
626                <1> ; Input:
627                <1> ; EDI = Pointer to buffer in which to place
628                <1> ; VbeInfoBlock structure
629                <1> ;
630                <1> ; AX = 4F00h
631                <1> ;
632                <1> ; Output:
633                <1> ; AX = VBE return status
634                <1> ; AX = 004Fh -> succeeded
635                <1> ; AX <> 004Fh -> failed
636                <1> ;
637                <1> ; Modified registers: eax (+ edi for kernel's own call)
638                <1> ;
639                <1> ; NOTE: TRDOS 386 v2 (v2.0.3) kernel calls this function
640                <1> ; during startup while cpu is in real mode
641                <1> ; (by using int 10h, 4F02h) and saves VbeInfoBlock at
642                <1> ; VBE3INFOBLOCK address (97E00h for TRDOS 386 v2.0.3).
643                <1> ;
644                <1> ; So...
645                <1> ; This VBE function is adjusted to return/move same info
646                <1> ; from VBE3INFOBLOCK to user's buffer in EDI.
647                <1> ; int 31h (int 10h) entrance
648                <1> ;
649 000018E5 21FF     <1>      and     edi, edi
650 000018E7 7423     <1>      jz      short vbe3_func_fail ; invalid buffer address !
651                <1> ;
652                <1> ; _vbe3_pmf_n_return_ctrl_info:
653                <1> ; or      edi, edi
654                <1> ; jnz     short _vbe_biosfn_return_ctrl_info
655                <1> ;
656                <1> ; ; this option may not be necessary - 12/12/2020
657                <1> ;
658                <1> ; ; edi = 0, kernel forces to get ctrl info again
659                <1> ; ; by using VESA VBE3 video bios's pmi
660                <1> ;
661                <1> ; ; push edi
662                <1> ; ; far call to VESA VBE3 PMI
663                <1> ; ; mov ax, 4F00h ; Return VBE Controller Info
664                <1> ; ; mov edi, VBE3INFOBLOCK-VBE3SAVERESTOREBLOCK
665                <1> ; ; ES selector base address = VBE3SAVERESTOREBLOCK
666                <1> ; ; call int10h_32bit_pmi
667                <1> ; ; pop edi
668                <1> ; ; mov edi, VBE3INFOBLOCK ; retn to the kernel sub
669                <1> ; ; cmp ax, 004Fh
670                <1> ; ; je      short vbe_ctrl_info_retn
671                <1> ; ; stc

```

```

672      <1>      ;retn
673      <1>
674      <1> _vbe_biosfn_return_ctrl_info:
675      <1>      push    edi
676      <1>      push    ecx
677      <1>      mov     esi, VBE3INFOBLOCK
678      <1>      ;mov     ecx, 512
679      <1>      ; 02/08/2022
680      <1>      sub     ecx, ecx
681      <1>      mov     ch, 2
682      <1>      ; ecx = 512
683      <1>      call   transfer_to_user_buffer
684      <1>      pop     ecx
685      <1>      pop     edi
686      <1>      jc      short vbe3_func_fail
687      <1>
688      <1>      xor     eax, eax
689      <1>      mov     al, 4Fh ; successful
690      <1> ;vbe_ctrl_info_retn:
691      <1>      retn
692      <1>
693      <1> vbe_biosfn_return_ctrl_info:
694      <1>      ; 12/12/2020
695      <1>
696      <1>      ; VBE function 4F00h - Return VBE Controller Information
697      <1>
698      <1>      ; Input:
699      <1>      ;      EDI = Pointer to buffer in which to place
700      <1>      ;      VbeInfoBlock structure
701      <1>
702      <1>      ;      AX = 4F00h
703      <1>      ; Output:
704      <1>      ;      AX = VBE return status
705      <1>      ;      AX = 004Fh -> succeeded
706      <1>      ;      AX <> 004Fh -> failed
707      <1>
708      <1>      ; Modified registers: eax
709      <1>
710      <1>      and     edi, edi
711      <1>      jz      short vbe3_func_fail ; invalid buffer addr !
712      <1>      jmp     short _vbe_biosfn_return_ctrl_info
713      <1>
714      <1> vbe3_pmfn_return_mode_info:
715      <1>      ; 02/08/2022 (TRDOS 386 kernel v2.0.5)
716      <1>      ; 21/12/2020
717      <1>      ; 12/12/2020
718      <1>
719      <1>      ; VBE function 4F01h - Return VBE Mode Information
720      <1>
721      <1>      ; Input:
722      <1>      ;      CX = Mode number (VESA VBE mode number)
723      <1>      ;      EDI = Pointer to ModeInfoBlock structure
724      <1>      ;      (256 bytes) -User's buffer address-
725      <1>      ;      EDI = 0 -> kernel call
726      <1>      ;      (do not transfer ModeInfoBlock
727      <1>      ;      to user's buffer address)
728      <1>      ;      AX = 4F01h
729      <1>      ; Output:
730      <1>      ;      AX = VBE return status
731      <1>      ;      AX = 004Fh -> succeeded
732      <1>      ;      AX <> 004Fh -> failed
733      <1>
734      <1>      ; Modified registers: eax, esi, edi
735      <1>
736      <1>      ; int 31h (int 10h) entrance
737      <1>
738      <1>      or      edi, edi
739      <1>      jnz     short _vbe3_pmfn_return_mode_info
740      <1>
741      <1> vbe3_func_fail:
742      <1>      mov     eax, 014Fh ; ah = 1 : Function call failed
743      <1>      ; al = 4Fh : Function is supported
744      <1>      retn
745      <1>
746      <1>      ; jump from '_vbe_biosfn_return_mode_info'
747      <1> _vbe3_pmfn_return_mode_info:
748      <1>      push    edi
749      <1>
750      <1>      ;; clear vbe3 'mode info block' buffer
751      <1>      ;push    ecx
752      <1>      ;xor     eax, eax
753      <1>      ;mov     ecx, 256/4
754      <1>      ;mov     edi, VBE3MODEINFOBLOCK
755      <1>      ;rep     stosd
756      <1>      ;pop     ecx
757      <1>
758      <1>      ; far call to VESA VBE3 PMI
759      <1>      ;mov     ax, 4F01h ; Return VBE Mode Information
760      <1>      mov     edi, VBE3MODEINFOBLOCK-VBE3SAVERESTOREBLOCK
761      <1>      ; ES selector base address = VBE3SAVERESTOREBLOCK
762      <1>      call   int10h_32bit_pmi
763      <1>
764      <1>      pop     edi
765      <1>
766      <1>      ;cmp     ax, 004Fh
767      <1>      ;jne     short vbe3_func_retn ; failed !
768      <1>
769      <1>      and     edi, edi
770      <1>      ;jz      short vbe3_func_success
771      <1>      ; 21/12/2020
772      <1>      jz      short vbe3_func_retn
773      <1>
774      <1>      cmp     ax, 004Fh
775      <1>      jne     short vbe3_func_retn ; failed !
776      <1>
777      <1>      push    ecx
778      <1>      mov     esi, VBE3MODEINFOBLOCK
779      <1>      ;mov     ecx, 256
780      <1>      ; 02/08/2022
781      <1>      sub     ecx, ecx
782      <1>      inc     ch
783      <1>      ; ecx = 256
784      <1>      call   transfer_to_user_buffer
785      <1>      pop     ecx
786      <1>      jc      short vbe3_func_fail
787      <1>
788      <1>      xor     eax, eax
789      <1>      mov     al, 4Fh ; successful
790      <1> vbe3_func_success:
791      <1> vbe3_func_retn:
792      <1>      retn
793      <1>
794      <1> vbe3_pmfn_return_current_mode:
795      <1>      ; 12/12/2020

```



```

796 <1> ;
797 <1> ; VBE function 4F03h - Return Current VBE Mode
798 <1> ;
799 <1> ; Input:
800 <1> ; none (AX = 4F03h)
801 <1> ; Output:
802 <1> ; AX = VBE return status
803 <1> ; AX = 004Fh -> succeeded
804 <1> ; AX <> 004Fh -> failed
805 <1> ; BX = Current VBE mode
806 <1> ; bit 0-13 = Mode number
807 <1> ; bit 14 = 0 windowed frame buffer model
808 <1> ; = 1 Linear frame buffer model
809 <1> ; bit 15
810 <1> ; = 0 Memory cleared at last mode set
811 <1> ; = 1 Memory not cleared at last mode set
812 <1> ;
813 <1> ; Modified registers: eax, ebx
814 <1> ;
815 <1> ; int 31h (int 10h) entrance
816 <1> ;
817 <1> ; far call to VESA VBE3 PMI
818 <1> ;
819 <1> ; mov eax, 4F03h ; Return Current VBE Mode
820 <1> vbe3_pmf_n_far_call:
821 <1> ; ES selector base address = VBE3SAVERESTOREBLOCK
822 <1> ; call int10h_32bit_pmi
823 <1> ; retn
824 0000193F E9EB000000 <1> jmp int10h_32bit_pmi
825 <1>
826 <1> vbe3_pmf_n_set_mode:
827 <1> ; 02/08/2022 (TRDOS 386 Kernel v2.0.5)
828 <1> ; 22/12/2020
829 <1> ; 21/12/2020
830 <1> ; 12/12/2020
831 <1> ;
832 <1> ; VBE function 4F02h - Set VBE Mode
833 <1> ;
834 <1> ; Input:
835 <1> ; BX = Desired Mode to set
836 <1> ; bit 0-13 = Mode number
837 <1> ; bit 14 = 0 windowed frame buffer model
838 <1> ; = 1 Linear frame buffer model
839 <1> ; bit 15
840 <1> ; = 0 Memory cleared at last mode set
841 <1> ; = 1 Memory not cleared at last mode set
842 <1> ; Output:
843 <1> ; AX = VBE return status
844 <1> ; AX = 004Fh -> succeeded
845 <1> ; AX <> 004Fh -> failed
846 <1> ;
847 <1> ; Modified registers: eax, ebx, esi (21/12/2020)
848 <1> ;
849 <1> ; int 31h (int 10h) entrance
850 <1> ;
851 <1> ; 22/12/2020 (VESA VBE3 feature)
852 <1> ; BX bit 11 is flag for
853 <1> ; user specified CRTC values for refresh rate
854 <1> ; 'test bh, 8'
855 <1> ; if bit 11 is set, EDI points to 'CRTCInfoBlock'
856 <1> ;
857 <1> ; 22/12/2020
858 <1> ; test bx for VBE video mode
859 <1> ; test bh, 1
860 <1> ; jnz short vbe3_sm_0
861 <1> ;
862 <1> ; use internal VBE mode set procedure
863 <1> ; for non-vbe (std VGA/CGA) modes
864 <1> ;
865 <1> ; (it is useful -as 4F02h function-
866 <1> ; to jump 'vbe_biosfn_set_mode'
867 <1> ; instead of direct jump to '_set_mode')
868 <1> ; ((eliminates additional push-pops and settings))
869 <1> ;
870 <1> ; jmp vbe_biosfn_set_mode
871 <1> ;
872 <1> vbe3_sm_0:
873 <1> ; push ds ; *
874 <1> ; push es ; **
875 <1> ; push ebp ; ***
876 <1> ; push esi ; ****
877 <1> ;
878 <1> ; Fit bx to VESA VBE2 type mode setting
879 <1> ; (bx bit 11 is used for custom CRTC values in VBE3)
880 <1> ; clear bit 9 to 11 (clear bh bit 1 to bit 3)
881 <1> ;
882 <1> ; 22/12/2020
883 00001944 57 <1> push edi ; *****
884 00001945 F6C708 <1> test bh, 8 ; Use user specified CRTC values
885 00001948 7530 <1> jnz short vbe3_sm_3 ; for refresh rate
886 <1> vbe3_sm_4:
887 0000194A 80E7C1 <1> and bh, 0C1h ; use bit 15, 14, 8 only (for bh)
888 <1> ; mov [vbe_mode_x], bh
889 <1> ;
890 0000194D 803D[1E680000]03 <1> cmp byte [CRT_MODE], 3 ; is current mode 03h ?
891 00001954 7509 <1> jne short vbe3_sm_1 ; no
892 <1> ;
893 <1> ; save mode 03h video pages and cursor positions
894 00001956 57 <1> push edi ; **|***
895 00001957 51 <1> push ecx ; *****
896 <1> ; push esi
897 <1> ;
898 00001958 E8CE040000 <1> call save_mode3_multiscreen
899 <1> ;
900 <1> ; pop esi
901 0000195D 59 <1> pop ecx ; *****
902 0000195E 5F <1> pop edi ; **|***
903 <1> vbe3_sm_1:
904 <1> ; ax = 4F02h
905 <1> ; bx = video mode number (vbe2 type)
906 0000195F E8CB000000 <1> call int10h_32bit_pmi
907 <1> ; call to far call to VBE3 PMI
908 <1> ;
909 00001964 6683F84F <1> cmp ax, 004Fh ; succeeded ?
910 00001968 750E <1> jne short vbe3_sm_2
911 <1> ; set current mode byte/sign to extended (SVGA) mode
912 0000196A C605[1E680000]FF <1> mov byte [CRT_MODE], 0FFh ; VESA VBE mode
913 <1> ; set current VBE mode word to bx input
914 00001971 66891D[0E9D0100] <1> mov [video_mode], bx
915 <1> vbe3_sm_2:
916 <1> ; 22/12/2020
917 00001978 5F <1> pop edi ; *****
918 00001979 C3 <1> retn
919 <1>

```

```

920 <1> vbe3_sm_3:
921 <1> ; 22/12/2020
922 <1> ; copy user's CRTInfoBlock to the buffer
923 0000197A 51 <1> push ecx
924 0000197B 89FE <1> mov esi, edi
925 0000197D BF807D0900 <1> mov edi, VBE3CRTINFOBLOCK
926 <1> ;mov ecx, 64
927 <1> ; 02/08/2022
928 00001982 29C9 <1> sub ecx, ecx
929 00001984 B140 <1> mov cl, 64
930 00001986 E8CAF10000 <1> call transfer_from_user_buffer
931 0000198B 59 <1> pop ecx
932 <1> ; set offset (es base addr is VBE3SAVERESTOREBLOCK)
933 0000198C 81EF00760900 <1> sub edi, VBE3SAVERESTOREBLOCK
934 00001992 EBB6 <1> jmp short vbe3_sm_4
935 <1>
936 <1> vesa_vbe3_pmi:
937 <1> ; 29/11/2023 - TRDOS 386 v2.0.7
938 <1> ; 12/12/2020
939 <1> ; 08/12/2020
940 <1> ; 07/12/2020
941 <1> ; 05/12/2020, 06/12/2020
942 <1> ; 03/12/2020, 04/12/2020
943 <1> ; 28/11/2020 (TRDOS 386 v2.0.3)
944 <1> ; VGA BIOS functions via
945 <1> ; VESA VBE3 Protected Mode Inface
946 <1> ; [vbe3] = 3 and [pmi32] > 0
947 <1>
948 <1> ; 04/12/2020
949 <1> ; Only 'set mode' will be redirected to vbe3 video bios
950 <1> ; (by setting mode 3 multiscreen parameters before and after)
951 <1>
952 <1> ; 29/11/2023 - TRDOS 386 v2.0.7
953 00001994 50 <1> push eax
954 00001995 0F20D8 <1> mov eax, cr3
955 00001998 870424 <1> xchg eax, [esp] ; **!**
956 0000199B 50 <1> push eax
957 <1> ;cmp esi, [k_page_dir]
958 <1> ;je short vesa_vbe3_pmi_x
959 0000199C A1[A0760100] <1> mov eax, [k_page_dir]
960 000019A1 0F22D8 <1> mov cr3, eax
961 <1> ;vesa_vbe3_pmi_x:
962 000019A4 58 <1> pop eax
963 <1>
964 <1> ; 06/12/2020
965 000019A5 20E4 <1> and ah, ah ; 0 = set mode function
966 000019A7 7402 <1> jz short vbe3_pmi_0
967 000019A9 EB7D <1> jmp vbe3_pmi_9
968 <1>
969 <1> vbe3_pmi_0:
970 <1> ; 07/12/2020
971 000019AB 88C4 <1> mov ah, al
972 000019AD 80E480 <1> and ah, 80h ; 0 or 80h
973 000019B0 30E0 <1> xor al, ah ; 8?h -> 0?h
974 <1>
975 <1> ;cmp al, 13h ; mode number above 13h is returned
976 <1> ;jna short vbe3_pmi_1
977 <1> ; ; back to default code due to uncertainty
978 <1> ; ; (>13h is not std for all svga bioses)
979 <1> ;jmp VGA_funcs_0
980 <1> vbe3_pmi_1:
981 <1> ; 07/12/2020
982 <1> ; Possible cases for VBE3 (PMI, ah=0) set mode:
983 <1> ; current mode > 07h and requested mode: any
984 <1> ; current mode <= 07h and requested mode > 07h
985 <1>
986 <1> ; 06/12/2020
987 000019B2 8825[37830100] <1> mov byte [noclearmem], ah ; 0 or 80h
988 <1> ; check current video mode if it is 03h
989 000019B8 803D[1E680000]03 <1> cmp byte [CRT_MODE], 3 ; current mode
990 000019BF 750B <1> jne short vbe3_pmi_3
991 <1> ; 07/12/2020
992 <1> ; check new video mode if it is 03h also
993 <1> ;cmp al, 3
994 <1> ;jne short vbe3_pmi_2
995 <1> ;mov byte [p_crt_mode], 80h ; clear video memory
996 <1> ;jmp short vbe3_pmi_5
997 <1> vbe3_pmi_2:
998 <1> ; case 1:
999 <1> ; Current mode is 03h and new mode is not 03h
1000 <1>
1001 <1> ; save video pages and cursor positions
1002 000019C1 56 <1> push esi
1003 000019C2 57 <1> push edi
1004 000019C3 51 <1> push ecx
1005 <1>
1006 <1> ; 12/12/2020
1007 <1> ;mov esi, 0B8000h ; mode 3 video memory
1008 <1> ;mov edi, 98000h ; backup location
1009 <1> ;mov ecx, (0B8000h-0B0000h)/4
1010 <1> ;rep movsd
1011 <1> ;
1012 <1> ;mov byte [p_crt_mode], 3 ; previous mode, backup sign
1013 <1> ;xchg cl, [ACTIVE_PAGE]
1014 <1> ;mov [p_crt_page], cl ; save as previous active page
1015 <1> ;
1016 <1> ;; save cursor positions
1017 <1> ;mov esi, CURSOR_POSN
1018 <1> ;mov edi, cursor_pposn ; cursor positions backup
1019 <1> ;mov cl, 4
1020 <1> ;rep movsd
1021 <1>
1022 <1> ; 12/12/2020
1023 000019C4 E862040000 <1> call save_mode3_multiscreen
1024 <1>
1025 000019C9 59 <1> pop ecx
1026 000019CA 5F <1> pop edi
1027 000019CB 5E <1> pop esi
1028 <1> vbe3_pmi_3:
1029 <1> ; 08/12/2020
1030 <1> ; 07/12/2020
1031 <1> ; case 3 or case 4
1032 000019CC A2[1E680000] <1> mov [CRT_MODE], al
1033 000019D1 3C03 <1> cmp al, 3
1034 000019D3 7407 <1> je short vbe3_pmi_4
1035 <1> ; case 4:
1036 <1> ; Current mode is not 03h and also new mode is not 03h
1037 000019D5 800D[35830100]80 <1> or byte [p_crt_mode], 80h ; 83h (case 1 -> case 4)
1038 <1> ;jmp short vbe3_pmi_5
1039 <1> vbe3_pmi_4:
1040 <1> ; case 3:
1041 <1> ;
1042 <1> ; Current mode is not 03h and new mode is 03h
1043 <1>

```

```

1044 <1> ;vbe3_pmi_5:
1045 <1> ;mov [CRT_MODE], al
1046 <1>
1047 000019DC E84E000000 <1> call int10h_32bit_pmi
1048 <1>
1049 000019E1 803D[1E680000]03 <1> cmp byte [CRT_MODE], 3 ; new video mode
1050 <1> ;jne vbe3_pmi_8 ; video mode <> 03h
1051 000019E8 7532 <1> jne short vbe3_pmi_8
1052 <1>
1053 <1> ;push eax ; 04/12/2020
1054 000019EA 53 <1> push ebx
1055 000019EB 51 <1> push ecx
1056 000019EC 52 <1> push edx
1057 000019ED 57 <1> push edi ; 03/12/2020
1058 <1>
1059 <1> ; 12/12/2020
1060 000019EE 56 <1> push esi
1061 000019EF E869040000 <1> call restore_mode3_multiscreen
1062 000019F4 5E <1> pop esi
1063 <1> ; AL = active video page
1064 <1>
1065 <1> ; 12/12/2020
1066 <1> ;mov al, [p_crt_page] ; previous mode 3 active page
1067 <1> ;
1068 <1> ;test byte [p_crt_mode], 7Fh ; 83h or 80h or 03h
1069 <1> ;jz short vbe3_pmi_6 ; do not restore video pages
1070 <1> ; ; clear current video page
1071 <1> ; ; case 3
1072 <1> ; ;
1073 <1> ; ; ([p_crt_mode] = 03h)
1074 <1> ; ;
1075 <1> ; ; New video mode is 3 while current video mode is not 3
1076 <1> ; ; (multi screen) video pages will be restored from 098000h
1077 <1> ; ;
1078 <1> ; ; restore video pages and cursor positions
1079 <1> ; ;
1080 <1> ;mov [ACTIVE_PAGE], al ; current mode 3 active page
1081 <1> ;
1082 <1> ;push esi
1083 <1> ;
1084 <1> ; ; restore video pages
1085 <1> ;mov esi, 98000h
1086 <1> ;mov edi, 0B8000h
1087 <1> ; ;mov ecx, 2000h
1088 <1> ;mov cx, 2000h ; 8K dwords (32K)
1089 <1> ;rep movsd
1090 <1> ;
1091 <1> ;mov [p_crt_mode], cl ; reset ('case 3' end condition)
1092 <1> ;
1093 <1> ; ; restore cursor positions
1094 <1> ;mov esi, cursor_pposn
1095 <1> ;mov edi, CURSOR_POSN
1096 <1> ; ;mov ecx, 4 ; restore all cursor positions (16 bytes)
1097 <1> ;mov cl, 4
1098 <1> ;rep movsd
1099 <1> ;
1100 <1> ;pop esi
1101 <1> ;
1102 <1> ; ; 07/12/2020
1103 <1> ; ; restore CRT_START according to ACTIVE_PAGE
1104 <1> ;mov [CRT_START], cx ; 0
1105 <1> ;
1106 <1> ; ; check active page and set it again if it is not 0
1107 <1> ;or al, al
1108 <1> ;jz short vbe3_pmi_7
1109 <1> ;
1110 <1> ;mov cl, al
1111 <1> ;vbe3_pmi_5:
1112 <1> ;add word [CRT_START], 4096
1113 <1> ;dec cl
1114 <1> ;jnz short vbe3_pmi_5
1115 <1>
1116 000019F5 B405 <1> mov ah, 05h ; set current video page
1117 <1> ; al = video page
1118 000019F7 E833000000 <1> call int10h_32bit_pmi
1119 <1>
1120 <1> ; check current cursor position & set it again if not 0,0
1121 <1> ;movzx ebx, byte [ACTIVE_PAGE]
1122 000019FC 0FB6D8 <1> movzx ebx, al
1123 000019FF D0E3 <1> shl bl, 1
1124 00001A01 81C3[BE760100] <1> add ebx, CURSOR_POSN
1125 00001A07 668B13 <1> mov dx, [ebx]
1126 00001A0A 6621D2 <1> and dx, dx
1127 00001A0D 7409 <1> jz short vbe3_pmi_7
1128 <1>
1129 <1> ;dx = cursor position (dl = column, dh = row)
1130 <1> ;mov bh, [ACTIVE_PAGE] ; 06/12/2020
1131 00001A0F 88C7 <1> mov bh, al
1132 00001A11 B402 <1> mov ah, 02h ; set cursor position
1133 00001A13 E817000000 <1> call int10h_32bit_pmi
1134 <1>
1135 <1> ;jmp short vbe3_pmi_7
1136 <1>
1137 <1> ;vbe3_pmi_6:
1138 <1> ; ; 07/12/2020
1139 <1> ; ; case 1, previous mode is 03h, current mode is 03h
1140 <1> ; ; 03/12/2020
1141 <1> ; cmp byte [noclearmem], 0
1142 <1> ; jna short vbe3_pmi_7 ; do not clear memory
1143 <1> ; ; clear video page
1144 <1> ; mov ecx, 1024 ; 4096/4
1145 <1> ; mov eax, 07200720h
1146 <1> ; mov edi, 0B8000h ; [crt_base]
1147 <1> ; add di, [CRT_START]
1148 <1> ; rep stosd ; FILL THE REGEN BUFFER WITH BLANKS
1149 <1>
1150 <1> vbe3_pmi_7:
1151 00001A18 5F <1> pop edi
1152 00001A19 5A <1> pop edx
1153 00001A1A 59 <1> pop ecx
1154 00001A1B 5B <1> pop ebx
1155 <1> ;pop eax ; 04/12/2020
1156 <1> vbe3_pmi_8:
1157 <1> ; 04/12/2020
1158 <1> ; (TRDOS 386 v2.0.3, INT 31h, ah=0 return)
1159 00001A1C 31C0 <1> xor eax, eax ; eax = 0 -> succesful
1160 <1> vesa_vbe3_pmi_retn:
1161 <1> ; 29/11/2023 - TRDOS 386 v2.0.7
1162 00001A1E 870424 <1> xchg eax, [esp] ; ***
1163 <1> ;cmp eax, [k_page_dir]
1164 <1> ;je short vesa_vbe3_pmi_retn_x
1165 00001A21 0F22D8 <1> mov cr3, eax
1166 <1> ;vesa_vbe3_pmi_retn_x:
1167 00001A24 58 <1> pop eax

```

```

1168      <1>      ;
1169 00001A25 07      <1>      pop     es    ; **
1170 00001A26 1F      <1>      pop     ds    ; *
1171 00001A27 CF      <1>      iretd
1172      <1>
1173      <1> vbe3_pmi_9:
1174      <1>      ; 06/12/2020
1175      <1>      ; cmp     ah, 10h ; Set/Get Palette Registers
1176      <1>      ; jnb     short vbe3_pmi_10
1177      <1>      ; 05/12/2020
1178 00001A28 E802000000      <1>      call    int10h_32bit_pmi
1179 00001A2D EBEF      <1>      jmp     short vesa_vbe3_pmi_retn
1180      <1>
1181      <1> ;vbe3_pmi_10:
1182      <1>      ; 06/12/2020
1183      <1>      ; jmp     VGA_funcs_0
1184      <1>
1185      <1> int10h_32bit_pmi:
1186      <1>      ; 03/12/2020
1187      <1>      ; 28/11/2020
1188      <1>      ; calling standard VGA Bios (INT 10h) functions
1189      <1>      ; by using 32 bit protected mode interface of
1190      <1>      ; VESA VBE3 Video Bios (with 'PMID' signature)
1191      <1>
1192      <1>      ; 03/12/2020
1193      <1>      ; eax, ebx, ecx, edx, edi will be used by vbios pmi
1194      <1>      ; (esi and ebp will not be used)
1195      <1>
1196      <1>      ; 03/12/2020
1197 00001A2F 56      <1>      push    esi
1198 00001A30 C1E010      <1>      shl     eax, 16 ; move function number (ax) to hw
1199 00001A33 8B35[149D0100]      <1>      mov     esi, [pmid_addr] ; linear address of
1200      <1>      ; PMInfo.Entrypoint pointer
1201      <1>      ; mov     ax, [esi+PMInfo.Entrypoint]
1202 00001A39 668B06      <1>      mov     ax, [esi]
1203 00001A3C C1C010      <1>      rol     eax, 16 ; move PM entry address to hw
1204      <1>      ; and move function number to lw (ax)
1205 00001A3F 5E      <1>      pop     esi
1206      <1>
1207      <1>      ; top of stack: ; (*)
1208      <1>      ; return (the caller) address of "int10h_32bit_pmi"
1209      <1>
1210 00001A40 E91FEDFFFF      <1>      jmp     _VBE3PMI_fcall ; will return to the caller (*)
1211      <1>
1212      <1> _vbe3_pmfns_srs_8:
1213      <1>      ; 17/01/2021
1214 00001A45 31DB      <1>      xor     ebx, ebx ; points to VBE3SAVERESTOREBLOCK
1215      <1>      _vbe3_pmfns_srs_9: ; 24/01/2021
1216 00001A47 66B8044F      <1>      mov     ax, 4F04h
1217 00001A4B EBE2      <1>      jmp     short int10h_32bit_pmi
1218      <1>
1219      <1> vbe3_pmfns_save_restore_state:
1220      <1>      ; 02/08/2022 (TRDOS 386 kernel v2.0.5)
1221      <1>      ; 24/01/2021
1222      <1>      ; 23/01/2021
1223      <1>      ; 16/01/2021 - 17/01/2021
1224      <1>      ; 14/01/2021
1225      <1>      ;
1226      <1>      ; VBE function 4F04h - Save/Restore Video State
1227      <1>      ;
1228      <1>      ; Input:
1229      <1>      ;     DL = sub function
1230      <1>      ;     CL = requested state
1231      <1>      ;     EBX = pointer to buffer (if DL<>00h)
1232      <1>      ;     AX = 4F04h
1233      <1>      ; Output:
1234      <1>      ;     AX = 004Fh (successful)
1235      <1>      ;     AH > 0 -> error
1236      <1>      ;     BX = Number of 64-byte blocks
1237      <1>      ;     to hold the state buffer (if DL=00h)
1238      <1>
1239      <1>      ; Modified registers: eax, ebx, esi, edi
1240      <1>
1241 00001A4D 21DB      <1>      and     ebx, ebx ; user's buffer address
1242 00001A4F 750A      <1>      jnz     short _vbe3_pmfns_save_restore_state
1243      <1>
1244 00001A51 08D2      <1>      or      dl, dl
1245 00001A53 740C      <1>      jz      short _vbe3_pmfns_srs_0
1246      <1>
1247      <1>      ; function failed
1248      <1>      ; mov     eax, 0100h
1249      <1>      ; sub     eax, eax
1250      <1>      ; inc     ah ; eax = 0100h
1251      <1>      ; retn
1252      <1>      ; 16/01/2021
1253      <1>      _vbe3_pmfns_srs_fail:
1254 00001A55 B84F010000      <1>      mov     eax, 014Fh ; ah = 1 : Function call failed
1255      <1>      ; al = 4Fh : Function is supported
1256      <1>      _vbe3_srs_retn:
1257 00001A5A C3      <1>      retn
1258      <1>
1259      <1> _vbe3_pmfns_save_restore_state:
1260 00001A5B 20D2      <1>      and     dl, dl
1261 00001A5D 7555      <1>      jnz     short _vbe3_pmfns_srs_2
1262      <1>      _vbe3_pmfns_srs:
1263 00001A5F 31DB      <1>      xor     ebx, ebx
1264      <1>      _vbe3_pmfns_srs_0:
1265      <1>      ; 24/01/2021
1266 00001A61 83F90F      <1>      cmp     ecx, 0Fh
1267      <1>      ; ja      short _vbe3_pmfns_srs_1
1268 00001A64 77EF      <1>      ja      short _vbe3_pmfns_srs_fail
1269      <1>
1270      <1>      ; !!! CLEAR CL BIT 2 !!!
1271      <1>      ; (when bit 2 is set, function causes cpu exception)
1272      <1>      ; BIOS data will not be saved and restored
1273      <1>      ; (to prevent protected mode page fault error)
1274 00001A66 80E1FD      <1>      and     cl, ~2 ; and cl, not 2
1275      <1>
1276      <1>      ; 24/01/2021
1277      <1>      ; mov     bl, 1
1278 00001A69 FEC3      <1>      inc     bl ; = 1
1279      <1>      ; shl     bx, cl
1280      <1>      ; 02/08/2022
1281 00001A6B D3E3      <1>      shl     ebx, cl
1282 00001A6D 66231D[189D0100]      <1>      and     bx, [vbe3stbsflags]
1283 00001A74 7415      <1>      jz      short _vbe3_pmfns_srs_1
1284      <1>      ; mov     bx, cx
1285 00001A76 89CB      <1>      mov     ebx, ecx ; <= 15
1286 00001A78 D0E3      <1>      shl     bl, 1 ; 0, 2, 8 .. 30
1287 00001A7A 668B9B[833B0000]      <1>      mov     bx, [vbestatebufsize+ebx]
1288 00001A81 89DF      <1>      mov     edi, ebx
1289      <1>      ; edi = state buffer size in bytes
1290      <1>      ; shr     bx, 6 ; / 64
1291      <1>      ; 02/08/2022

```

```

1292 00001A83 C1EB06      <1>      shr     ebx, 6
1293                    <1>      ;mov     ax, 4Fh
1294 00001A86 28E4      <1>      sub     ah, ah
1295 00001A88 B04F      <1>      mov     al, 4Fh
1296 00001A8A C3        <1>      retn
1297                    <1>      _vbe3_pmfnsrs_1:
1298                    <1>      ; ax = 4F04h
1299                    <1>      ; call    int10h_32bit_pmi
1300                    <1>      ; 24/01/2021
1301                    <1>      ; call    _vbe3_pmfnsrs_8
1302                    <1>      ; ebx = 0
1303 00001A8B E8B7FFFFFF <1>      call    _vbe3_pmfnsrs_9
1304 00001A90 6683F84F <1>      cmp     ax, 004Fh
1305 00001A94 75C4      <1>      jne     short _vbe3_srs_retn
1306                    <1>      ; 24/01/2021
1307                    <1>      ; cmp     ecx, 0Fh
1308                    <1>      ; ja      short _vbe3_srs_retn
1309                    <1>      ; 24/01/2021
1310                    <1>      ; mov     ax, 1
1311 00001A96 B001      <1>      mov     al, 1
1312                    <1>      ; shl     ax, cl
1313                    <1>      ; 02/08/2022
1314 00001A98 D3E0      <1>      shl     eax, cl
1315 00001A9A 660905[189D0100] <1>      or      [vbe3stbsflags], ax ; set flag for state option
1316                    <1>      ; 23/01/2021
1317 00001AA1 89DF      <1>      mov     edi, ebx
1318 00001AA3 89C8      <1>      mov     eax, ecx
1319 00001AA5 D0E0      <1>      shl     al, 1
1320                    <1>      ; shl     di, 6 ; * 64
1321                    <1>      ; 02/08/2022
1322 00001AA7 C1E706      <1>      shl     edi, 6
1323 00001AAA 6689B8[833B0000] <1>      mov     [vbestatebufsize+eax], di
1324                    <1>      ; save buf size for option
1325                    <1>      ; xchg     edi, ebx
1326                    <1>      ; edi = state buffer size in bytes
1327 00001AB1 B04F      <1>      mov     al, 4Fh
1328 00001AB3 C3        <1>      retn
1329                    <1>
1330                    <1>      _vbe3_pmfnsrs_2:
1331                    <1>      ; 24/01/2021
1332                    <1>      ; !!! CLEAR CL BIT 2 !!!
1333                    <1>      ; (when bit 2 is set, function causes cpu exception)
1334                    <1>      ; BIOS data will not be saved and restored
1335                    <1>      ; (to prevent protected mode page fault error)
1336                    <1>
1337 00001AB4 F6C1FD      <1>      test    cl, ~2 ; test cl, not 2
1338 00001AB7 749C      <1>      jz      short _vbe3_pmfnsrs_fail
1339                    <1>
1340 00001AB9 80FA02      <1>      cmp     dl, 2
1341 00001ABC 7748      <1>      ja      short _vbe3_pmfnsrs_5
1342                    <1>
1343                    <1>      ; and     cl, ~2 ; and cl, not 2
1344                    <1>
1345 00001ABE 53        <1>      push     ebx ; * ; buffer address
1346                    <1>      ; save or restore state
1347                    <1>      ; (get required buffer size at first)
1348 00001ABF 52        <1>      push     edx ; **
1349 00001AC0 28D2      <1>      sub     dl, dl ; 0
1350 00001AC2 E898FFFFFF <1>      call    _vbe3_pmfnsrs
1351 00001AC7 5A        <1>      pop      edx ; **
1352                    <1>      ; 24/01/2021
1353 00001AC8 5B        <1>      pop     ebx ; *
1354 00001AC9 08E4      <1>      or      ah, ah
1355 00001ACB 7538      <1>      jnz     short _vbe3_pmfnsrs_4 ; error
1356                    <1>
1357                    <1>      ; edi = buffer size in bytes
1358 00001ACD 81FF00080000 <1>      cmp     edi, 2048
1359 00001AD3 772B      <1>      ja      short _vbe3_pmfnsrs_3
1360                    <1>
1361 00001AD5 80FA01      <1>      cmp     dl, 1
1362 00001AD8 7531      <1>      jne     short _vbe3_pmfnsrs_6 ; restore state
1363                    <1>
1364                    <1>      ; save video state
1365                    <1>      ; xor     ebx, ebx ; points to VBE3SAVERESTOREBLOCK
1366                    <1>      ; mov     ax, 4F04h
1367                    <1>      ; call    int10h_32bit_pmi
1368                    <1>
1369                    <1>      ; 24/01/2021
1370 00001ADA E842000000 <1>      call    _vbe3_pmfnsrs_7
1371                    <1>
1372 00001ADF 6683F84F <1>      cmp     ax, 004Fh
1373 00001AE3 7520      <1>      jne     short _vbe3_pmfnsrs_4
1374                    <1>
1375 00001AE5 09DB      <1>      or      ebx, ebx ; kernel ('sysvideo') ?
1376 00001AE7 741C      <1>      jz      short _vbe3_pmfnsrs_4 ; yes
1377                    <1>
1378                    <1>      ; the caller is user
1379 00001AE9 51        <1>      push     ecx ; *
1380 00001AEA 89F9      <1>      mov     ecx, edi ; state buffer size
1381 00001AEC BE00760900 <1>      mov     esi, VBE3SAVERESTOREBLOCK ; source
1382                    <1>      ; (vbe3 pmi buff)
1383 00001AF1 89DF      <1>      mov     edi, ebx ; destination (user buff)
1384 00001AF3 E813F00000 <1>      call    transfer_to_user_buffer
1385 00001AF8 59        <1>      pop      ecx ; *
1386 00001AF9 7205      <1>      jc      short _vbe3_pmfnsrs_3
1387                    <1>
1388 00001AFB 29C0      <1>      sub     eax, eax
1389 00001AFD B04F      <1>      mov     al, 4Fh
1390 00001AFF C3        <1>      retn
1391                    <1>
1392                    <1>      ; 24/01/2021
1393                    <1>      _vbe3_pmfnsrs_3:
1394 00001B00 B84F010000 <1>      mov     eax, 014Fh
1395                    <1>      _vbe3_pmfnsrs_4:
1396                    <1>      retn
1397                    <1>      _vbe3_pmfnsrs_5:
1398                    <1>      xor     eax, eax
1399 00001B08 FEC4      <1>      inc     ah
1400                    <1>      ; eax = 0100h, function is not supported
1401 00001B0A C3        <1>      retn
1402                    <1>
1403                    <1>      _vbe3_pmfnsrs_6:
1404                    <1>      ; restore video state
1405                    <1>      ; 24/01/2021
1406                    <1>      ; pop     ebx ; *
1407                    <1>      ; 23/01/2021
1408 00001B0B 09DB      <1>      or      ebx, ebx ; 0 ?
1409 00001B0D 7412      <1>      jz      short _vbe3_pmfnsrs_7 ; 'sysvideo' call
1410                    <1>      ; 24/01/2021
1411                    <1>      ; jz      _vbe3_pmfnsrs_8
1412 00001B0F 89DE      <1>      mov     esi, ebx
1413                    <1>      ; esi = user's video state buffer
1414 00001B11 51        <1>      push     ecx ; *
1415 00001B12 89F9      <1>      mov     ecx, edi ; state buffer size

```

```

1416 00001B14 BF00760900 <1> mov edi, VBE3SAVERESTOREBLOCK ; destination
1417 <1> ; (vbe3 pmi buff)
1418 <1> ;mov esi, ebx ; source (user buff)
1419 00001B19 E837F00000 <1> call transfer_from_user_buffer
1420 00001B1E 59 <1> pop ecx ; *
1421 00001B1F 72DF <1> jc short _vbe3_pmfns_srs_3
1422 <1> _vbe3_pmfns_srs_7:
1423 00001B21 53 <1> push ebx ; *
1424 <1> ; restore video state
1425 <1> ;xor ebx, ebx ; points to VBE3SAVERESTOREBLOCK
1426 <1> ;mov ax, 4F04h
1427 <1> ;call int10h_32bit_pmi
1428 <1> ; 17/01/2021
1429 00001B22 E81EFFFF <1> call _vbe3_pmfns_srs_8
1430 00001B27 5B <1> pop ebx ; *
1431 00001B28 C3 <1> retn
1432 <1>
1433 <1> VIDEO_STATE:
1434 <1> ; 26/06/2016
1435 <1> ; 12/05/2016
1436 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1437 <1>
1438 <1> ;-----
1439 <1> ; VIDEO STATE
1440 <1> ; RETURNS THE CURRENT VIDEO STATE IN AX
1441 <1> ; AH = NUMBER OF COLUMNS ON THE SCREEN
1442 <1> ; AL = CURRENT VIDEO MODE
1443 <1> ; BH = CURRENT ACTIVE PAGE
1444 <1> ;-----
1445 <1>
1446 00001B29 8A25[20680000] <1> mov ah, [CRT_COLS] ; GET NUMBER OF COLUMNS
1447 00001B2F A0[1E680000] <1> mov al, [CRT_MODE] ; CURRENT MODE
1448 <1> ;movzx esi, al
1449 <1> ;mov ah, [esi+M6]
1450 <1> ; BH = active page
1451 00001B34 8A3D[CE760100] <1> mov bh, [ACTIVE_PAGE] ; GET CURRENT ACTIVE PAGE
1452 00001B3A FA <1> cli ; 02/01/2017
1453 00001B3B 5D <1> pop ebp ; RECOVER REGISTERS
1454 00001B3C 5F <1> pop edi
1455 00001B3D 5E <1> pop esi
1456 00001B3E 59 <1> pop ecx ; DISCARD SAVED BX
1457 00001B3F EB41 <1> jmp short M15 ; RETURN TO CALLER
1458 <1>
1459 <1> set_mode_ncm:
1460 <1> ; 17/11/2020 (TRDOS 386 v2.0.3)
1461 <1> ; 04/07/2016 - TRDOS 386 (TRDOS v2.0)
1462 <1> ; set mode without clearing the video memory
1463 <1> ; (only for graphics modes)
1464 <1>
1465 <1> ;cmp al, 7 ; IBM PC CGA modes
1466 <1> ;jna short SET_MODE ; normal function (clear)
1467 <1> ; do not clear memory
1468 <1> ;mov [noclearmem], al ; > 0
1469 <1> ;mov byte [noclearmem], 80h ; 17/11/2020
1470 <1> ;call _set_mode
1471 <1> ;mov byte [noclearmem], 0
1472 <1> ;jmp short VIDEO_RETURN
1473 <1>
1474 <1> ; 17/11/2020 (TRDOS v2.0.3)
1475 00001B41 0C80 <1> or al, 80h ; not clear memory option
1476 <1>
1477 <1> ; 05/12/2020
1478 <1> ; 27/11/2020
1479 <1> ; 17/11/2020
1480 <1> ; 08/08/2016, 10/08/2016
1481 <1> ; 29/07/2016, 30/07/2016
1482 <1> ; 25/07/2016, 26/07/2016, 27/07/2016
1483 <1> ; 02/07/2016, 18/07/2016, 23/07/2016
1484 <1> ; 24/06/2016, 26/06/2016
1485 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
1486 <1> SET_MODE:
1487 <1> ; For 32 bit TRDOS and Retro UNIX 386:
1488 <1> ; valid video mode: 03h only!
1489 <1> ; (VGA modes will be selected with another routine)
1490 <1> ;
1491 <1> ; set_txt_mode ; 80*25 (16 fore colors, 8 back colors)
1492 <1>
1493 <1> ; 27/11/2020
1494 <1>
1495 <1> ; Check if current mode is
1496 <1> ; Bochs/Plex86 VBE graphics mode
1497 00001B43 803D[1E680000]FF <1> cmp byte [CRT_MODE], 0FFh ; VESA VBE graphics mode
1498 00001B4A 7220 <1> jb short _set_mode_ ; signature
1499 <1> ; VBE mode number is in
1500 <1> ; [video_mode] bit 0to8
1501 00001B4C 88C3 <1> mov bl, al ; save video mode
1502 00001B4E E87C230000 <1> call dispi_get_enable
1503 00001B53 50 <1> push eax ; save current VBE dispi status
1504 <1> ; Disable Bochs/Plex86 VBE dispi
1505 <1> ;mov ax, 0 ; VBE_DISPI_DISABLED
1506 00001B54 31C0 <1> xor eax, eax ; 0
1507 00001B56 E849230000 <1> call dispi_set_enable
1508 00001B5B 88D8 <1> mov al, bl ; restore video mode
1509 00001B5D E827000000 <1> call _set_mode
1510 00001B62 58 <1> pop eax ; restore current VBE dispi status
1511 00001B63 7313 <1> jnc short VIDEO_RETURN
1512 <1> ; ! unimplemented or invalid video mode number !
1513 <1> ; VBE dispi must be enabled again
1514 <1> ; (return to run on current VBE graphics mode)
1515 <1> ;mov al, [video_mode+1] ; bit 8 to 15
1516 <1> ;and al, 0C0h ; isolate bit 14 and bit 15
1517 <1> ;or al, 1 ; VBE_DISPI_ENABLED
1518 00001B65 E83A230000 <1> call dispi_set_enable
1519 00001B6A EB07 <1> jmp short _video_func_err
1520 <1>
1521 <1> _set_mode_:
1522 <1> ; VGA bios (non-VBE) 'setmode' procedure
1523 <1>
1524 <1> ; 26/11/2020 (TRDOS v2.0.3)
1525 <1>
1526 <1> ;-----
1527 <1> ; SET MODE :
1528 <1> ; THIS ROUTINE INITIALIZES THE ATTACHMENT TO :
1529 <1> ; THE SELECTED MODE, THE SCREEN IS BLANKED. :
1530 <1> ; INPUT :
1531 <1> ; (AL) - MODE SELECTED (RANGE 0-7) :
1532 <1> ; OUTPUT :
1533 <1> ; NONE :
1534 <1> ;-----
1535 <1>
1536 00001B6C E818000000 <1> call _set_mode ; 24/06/2016 (set_txt_mode)
1537 <1> ; 26/11/2020
1538 00001B71 7305 <1> jnc short VIDEO_RETURN
1539 <1>

```

```

1540      ; 26/11/2020
1541      _video_func_err:
1542      xor     eax, eax ; function call failed
1543      dec     eax ; 0FFFFFFFFh ; - 1
1544      jmp     short _video_return
1545
1546      ; 12/05/2016
1547      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1548
1549      ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
1550
1551      VIDEO_RETURN:
1552      mov     eax, [video_eax] ; 12/05/2016
1553      _video_return:
1554      cli ; 02/01/2017
1555      pop     ebp ; ***** ; 26/11/2020
1556      pop     edi ; *****
1557      pop     esi ; *****
1558      pop     ebx ; *****
1559      M15: ; VIDEO_RETURN_C
1560      ; 15/01/2017
1561      ; 02/01/2017
1562      ; mov     byte [intflg], 0
1563
1564      pop     ecx ; **** ; 26/11/2020
1565      pop     edx ; ***
1566      pop     ds ; **
1567      pop     es ; * ; RECOVER SEGMENTS
1568      iretd   ; ALL DONE
1569
1570      set_txt_mode:
1571
1572      ; 29/07/2016
1573      ; 27/06/2016
1574      mov     al, 3 ; 26/11/2020 (bit 7 = 0)
1575
1576      ; 17/11/2020 (TRDOS v2.0.3)
1577      mov     byte [noclearmem], 0
1578
1579      ; 04/08/2022
1580      ; 02/08/2022 (TRDOS 386 kernel v2.0.5)
1581      ; 12/04/2021
1582      ; 10/08/2016
1583      ; 08/08/2016
1584      ; 30/07/2016
1585      ; 29/07/2016
1586      ; 25/07/2016 - 26/07/2016 - 27/07/2016
1587      ; 07/07/2016 - 18/07/2016 - 23/07/2016
1588      ; 02/07/2016 - 03/07/2016 - 04/07/2016
1589      ; 26/06/2016
1590      ; 24/06/2016 (set_txt_mode -> _set_mode)
1591      ; 17/06/2016
1592      ; 29/05/2016
1593      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
1594
1595      _set_mode:
1596      ; 12/12/2020
1597      ; 26/11/2020
1598      ; call from 'biosfn_set_video_mode'
1599      ; (bochs/plex86 video bios code)
1600      ; call from 'SET_MODE'
1601      ; (TRDOS 386 v2 default, IBM PC/AT rom bios code)
1602      ; continue from 'set_txt_mode'
1603
1604      ; INPUT:
1605      ;     al = VGA video mode
1606      ; RETURN:
1607      ;     cf = 1 -> video mode not implemented
1608      ;     cf = 0 -> OK
1609
1610      ; Modified registers: eax, bx, ecx, esi, edi, (ebp)
1611
1612      ; 17/11/2020 (TRDOS v2.0.3)
1613      ; no clear memory option
1614      ; (from mode number byte bit 7)
1615      mov     ah, al
1616      and     ah, 80h
1617      mov     [noclearmem], al
1618      mov     [noclearmem], ah
1619      and     al, 7Fh ; clear bit 7
1620      xor     [noclearmem], al ; clear bit 0 to 6
1621      ; 26/11/2020
1622      xor     al, ah ; and al, 7Fh
1623
1624      ; 19/11/2020
1625
1626      ; Video mode 03h action principle:
1627
1628      ; for case 1:
1629      ; Current mode is 03h and next/requested mode is not 03h
1630      ; - save mode (set mode 03h flag)
1631      ; - save 8 video pages (which are will be restored)
1632      ; - save active page number (which will be reactivated)
1633      ; - set active page to 0 always (no multi screen)
1634      ; - save 8 cursor positions (which will be restored)
1635      ; - use 'noclearmem' option
1636      ; [p_crt_mode] = 0 -> 03h
1637
1638      ; for case 2:
1639      ; Current mode is 03h and next/requested mode is also 03h
1640      ; - clear active video page if 'noclearmem' is not set
1641      ; [p_crt_mode] = 0 -> 80h -> 0
1642
1643      ; for case 3:
1644      ; Current mode is not 03h and next/requested mode is 03h
1645      ; - restore video pages (8 video pages were saved)
1646      ; - restore active page number (which were saved)
1647      ; - restore 8 cursor positions (which were saved)
1648      ; - reset/clear mode 03h flag
1649      ; [p_crt_mode] = 03h -> 0
1650
1651      ; for case 4:
1652      ; Current mode is not 03h and next/requested mode is not 03h
1653      ; - use 'noclearmem' option
1654      ; - set active page to 0 always
1655      ; [p_crt_mode] = 03h -> 83h -> 03h
1656
1657      ; initial (boot time) values:
1658      ; [p_crt_mode] = 0 ("there isn't a page backup, yet")
1659      ; [CRT_MODE] = 3 (kernel's starting mode)
1660
1661      ; 26/11/2020
1662      cmp     al, 03h ; mode 3, 80x25 text, 16 colors
1663      jne     short _sm_0 ; (default mode for TRDOS 386 mainprog)

```

```

1664 <1>
1665 <1> ; case 2 or case 3
1666 <1>
1667 <1> ; check current video mode if it is 03h
1668 00001B9A 08E4 <1> or ah, ah ; 80h or 0 ('noclearmem' option)
1669 00001B9C 7521 <1> jnz short _sm_1 ; do not clear display page
1670 <1>
1671 <1> ; 26/11/2020
1672 <1> ; Note:
1673 <1> ; [CRT_MODE] = 0FFh for VESA VBE video modes
1674 <1> ; [video_mode] = standard VGA and VESA VBE video modes
1675 <1>
1676 00001B9E 3805[1E680000] <1> cmp [CRT_MODE], al ; 03h
1677 00001BA4 7520 <1> jne short _sm_2 ; case 3 ([p_crt_mode] = 03h)
1678 <1>
1679 <1> ; case 2
1680 <1>
1681 <1> ; [p_crt_mode] = 0
1682 <1>
1683 <1> ; 19/11/2020
1684 <1> ; If '_set_mode' procedure is called for video mode 3
1685 <1> ; while video mode is 3, video page will be cleared
1686 <1> ; and cursor position of video page will be reset.
1687 <1>
1688 <1> ; clear display page
1689 00001BA6 C605[35830100]80 <1> mov byte [p_crt_mode], 80h ; clear page sign
1690 00001BAD EB1C <1> jmp short _sm_3 ; bypass save video page routine
1691 <1> _sm_0:
1692 <1> ; case 1 or case 4
1693 <1>
1694 <1> ; 05/12/2020
1695 00001BAF 803D[1E680000]03 <1> cmp byte [CRT_MODE], 3 ; is current mode 03h?
1696 00001BB6 7507 <1> jne short _sm_1 ; case 4 ; [p_crt_mode] = 03h
1697 <1>
1698 <1> ; case 1
1699 <1> ; [p_crt_mode] = 0
1700 <1>
1701 <1> ; 19/11/2020
1702 <1> ; If '_set_mode' procedure is called for a video mode
1703 <1> ; except video mode 3 while current video mode
1704 <1> ; is 3, all video pages of mode 3 will be copied
1705 <1> ; to 98000h address as backup, before mode change.
1706 <1>
1707 <1> _sm_save_pm:
1708 <1> ; 12/12/2020
1709 <1> ; 03/07/2016
1710 <1> ; save video pages
1711 <1> ; mov esi, 0B8000h
1712 <1> ; mov edi, 98000h ; 30/07/2016
1713 <1> ; mov ecx, (0B8000h-0B0000h)/4
1714 <1> ; rep movsd
1715 <1>
1716 <1> ; mov byte [p_crt_mode], 3 ; previous mode, backup sign
1717 <1> ; 26/11/2020
1718 <1> ; xchg cl, [ACTIVE_PAGE]
1719 <1> ; mov [p_crt_page], cl ; save as previous active page
1720 <1> ;
1721 <1> ; save cursor positions
1722 <1> ; mov esi, CURSOR_POSN
1723 <1> ; mov edi, cursor_pposn ; cursor positions backup
1724 <1> ; mov cl, 4
1725 <1> ; rep movsd
1726 <1>
1727 <1> ; 12/12/2020
1728 00001BB8 E86E020000 <1> call save_mode3_multiscreen
1729 <1>
1730 <1> ; 29/07/2016
1731 <1> ; mov [ACTIVE_PAGE], cl ; 0
1732 <1> ; xchg cl, [ACTIVE_PAGE]
1733 <1> ; mov [p_crt_page], cl ; previous page (for mode 3)
1734 <1>
1735 <1> ; [ACTIVE_PAGE] = 0
1736 <1>
1737 00001BBD EB07 <1> jmp short _sm_2 ; case 1 - 19/11/2020
1738 <1> _sm_1:
1739 <1> ; 26/11/2020
1740 <1> ; 19/11/2020
1741 <1>
1742 <1> ; case 4
1743 00001BBF 800D[35830100]80 <1> or byte [p_crt_mode], 80h
1744 <1> ; here [p_crt_mode] must be 83h
1745 <1> ; (for case 4)
1746 <1> ; (because video mode 03h
1747 <1> ; was changed before as in case 1)
1748 <1>
1749 <1> _sm_2: ; case 4 (jump to _sm_2) - 19/11/2020
1750 <1>
1751 <1> ; 19/11/2020
1752 <1> ; case 3
1753 <1> ; If '_set_mode' procedure is called for video mode 3
1754 <1> ; while video mode is not 3 and if there is video
1755 <1> ; page backup for video mode 3, all (of 8) mode 3
1756 <1> ; video pages will be restored from 98000h.
1757 <1>
1758 00001BC6 A2[1E680000] <1> mov [CRT_MODE], al ; save mode in global variable
1759 <1> _sm_3:
1760 <1> ; 04/08/2022 (TRDOS 386 v2.0.5)
1761 <1> ; 30/07/2016
1762 <1> ; 26/07/2016
1763 <1> ; 25/07/2016
1764 <1> ; set_mode_vga:
1765 <1> ; 18/07/2016
1766 <1> ; 14/07/2016
1767 <1> ; 09/07/2016
1768 <1> ; 04/07/2016
1769 <1> ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
1770 <1> ; /// video mode 13h ///
1771 <1> ; derived from 'Plex86/Bochs VGABios' source code
1772 <1> ; vgabios-0.7a (2011)
1773 <1> ; by the LGPL VGABios developers Team (2001-2008)
1774 <1> ; 'vgabios.c', 'vgatables.h'
1775 <1> ;
1776 <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code
1777 <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
1778 <1> ;
1779 00001BCB 88C4 <1> mov ah, al
1780 00001BCD B910000000 <1> mov ecx, vga_mode_count
1781 00001BD2 BE[3A680000] <1> mov esi, vga_modes
1782 00001BD7 31DB <1> xor ebx, ebx
1783 <1> _sm_4:
1784 00001BD9 AC <1> lodsb
1785 00001BDA 38C4 <1> cmp ah, al
1786 00001BDC 7406 <1> je short _sm_5
1787 00001BDE FEC3 <1> inc bl

```



```

1788 00001BE0 E2F7      <1>      loop    _sm_4
1789                  <1>
1790                  <1>      ; UNIMPLEMENTED VIDEO MODE !
1791                  <1>      ;xor     eax, eax
1792                  <1>      ;mov     [video_eax], eax ; 0
1793                  <1>
1794                  <1>      ; 26/11/2020
1795 00001BE2 F9          <1>      stc      ; unimplemented video mode ! (cf=1)
1796                  <1>
1797 00001BE3 C3          <1>      retn
1798                  <1>
1799                  <1>      ;-----      EBX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
1800                  <1>
1801                  <1>      _sm_5:      ; 25/07/2016
1802                  <1>      ;mov     esi, ebx
1803                  <1>      ;add     esi, vga_memmodel
1804                  <1>      ;mov     al, [esi]
1805                  <1>      ; 19/11/2020
1806 00001BE4 8A83[8A680000] <1>      mov     al, [ebx+vga_memmodel]
1807 00001BEA A2[4E830100] <1>      mov     [VGA_MTYPE], al
1808                  <1>
1809 00001BEF 89DF          <1>      mov     edi, ebx
1810 00001BF1 81C7[9A680000] <1>      add     edi, vga_dac_s
1811 00001BF7 C0E302       <1>      shl     bl, 2 ; byte -> dword
1812 00001BFA 81C3[4A680000] <1>      add     ebx, vga_mode_tbl_ptr
1813                  <1>
1814                  <1>      ;mov     dword [VGA_BASE], 0B8000h
1815                  <1>      ;cmp     ah, 0Dh ; [CRT_MODE]
1816                  <1>      ;jb      short M9
1817                  <1>      ;mov     dword [VGA_BASE], 0A0000h
1818                  <1>      ;M9:
1819 00001C00 8B33          <1>      mov     esi, [ebx]
1820 00001C02 89F3          <1>      mov     ebx, esi
1821 00001C04 83C614       <1>      add     esi, vga_p_cm_pos ; ebx + 20
1822 00001C07 668B06       <1>      mov     ax, [esi] ; get the cursor mode from the table
1823 00001C0A 66A3[37680000] <1>      mov     [CURSOR_MODE], ax ; save cursor mode (initial value)
1824                  <1>      ; al = 6, ah = 7
1825                  <1>      ; al = 0Dh, ah = 0Eh ; 25/07/2016
1826 00001C10 E893020000   <1>      call    cursor_shape_fix
1827                  <1>      ; al = 14, ah = 15 (If [CHAR_HEIGHT] = 16)
1828 00001C15 668906       <1>      mov     [esi], ax
1829                  <1>
1830 00001C18 56           <1>      push    esi ; *
1831                  <1>
1832                  <1>      ; 17/04/2021
1833 00001C19 B603          <1>      mov     dh, 03h
1834                  <1>
1835 00001C1B 8A25[25680000] <1>      mov     ah, [VGA_MODESET_CTL]
1836 00001C21 80E408       <1>      and     ah, 8 ; default palette loading ?
1837 00001C24 7520          <1>      jnz     short _sm_6
1838                  <1>      ;mov     dx, 3C6h ; VGAREG_PEL_MASK (DAC mask register)
1839                  <1>      ; 17/04/2021
1840 00001C26 B2C6          <1>      mov     dl, 0C6h
1841 00001C28 B0FF          <1>      mov     al, 0FFh ; PEL mask
1842 00001C2A EE           <1>      out     dx, al
1843 00001C2B 8A27          <1>      mov     ah, [edi] ; DAC model (selection number)
1844 00001C2D E8F80F0000   <1>      call    load_dac_palette
1845                  <1>      ; ecx = 0
1846 00001C32 F605[25680000]02 <1>      test    byte [VGA_MODESET_CTL], 2 ; gray scale summing
1847 00001C39 740B          <1>      jz      short _sm_6
1848 00001C3B 53           <1>      push    ebx
1849 00001C3C 29DB          <1>      sub     ebx, ebx ; sub bl, bl
1850                  <1>      ;mov     cx, 256
1851                  <1>      ; 02/08/2022
1852                  <1>      ;sub     ecx, ecx
1853                  <1>      ; ecx = 0
1854 00001C3E FEC5          <1>      inc     ch
1855                  <1>      ; ecx = 256
1856 00001C40 E837100000   <1>      call    gray_scale_summing
1857 00001C45 5B           <1>      pop     ebx
1858                  <1>      _sm_6:
1859                  <1>      ; Reset Attribute Ctl flip-flop
1860                  <1>      ;mov     dx, 3DAh ; VGAREG_ACTL_RESET
1861                  <1>      ; 17/03/2021
1862 00001C46 B2DA          <1>      mov     dl, 0DAh ; dx = 3DAh
1863 00001C48 EC           <1>      in     al, dx
1864                  <1>      ; Set Attribute Ctl
1865 00001C49 89DE          <1>      mov     esi, ebx ; addr of params tbl for selected mode
1866 00001C4B 83C623       <1>      add     esi, 35 ; actl regs
1867 00001C4E 30E4          <1>      xor     ah, ah ; 0
1868                  <1>      ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
1869                  <1>      ; 17/04/2021
1870 00001C50 B2C0          <1>      mov     dl, 0C0h
1871                  <1>      _sm_7:
1872 00001C52 88E0          <1>      mov     al, ah
1873 00001C54 EE           <1>      out     dx, al ; index
1874 00001C55 AC           <1>      lodsb
1875                  <1>      ; DX = 3C0h = VGAREG_ACTL_WRITE_DATA
1876 00001C56 EE           <1>      out     dx, al ; value
1877 00001C57 FEC4          <1>      inc     ah
1878 00001C59 80FC14       <1>      cmp     ah, 20 ; number of actl registers
1879 00001C5C 72F4          <1>      jb      short _sm_7
1880                  <1>
1881 00001C5E 88E0          <1>      mov     al, ah ; 20
1882 00001C60 EE           <1>      out     dx, al ; index
1883 00001C61 28C0          <1>      sub     al, al ; 0
1884 00001C63 EE           <1>      out     dx, al ; value
1885                  <1>
1886                  <1>      ; Set Sequencer Ctl
1887 00001C64 89DE          <1>      mov     esi, ebx ; addr of params tbl for selected mode
1888 00001C66 83C605       <1>      add     esi, 5 ; sequ regs
1889                  <1>
1890                  <1>      ;mov     dx, 3C4h ; VGAREG_SEQU_ADDRESS
1891                  <1>      ; 17/04/2021
1892 00001C69 B2C4          <1>      mov     dl, 0C4h
1893 00001C6B EE           <1>      out     dx, al ; 0
1894                  <1>      ;inc     dx ; 3C5h ; VGAREG_SEQU_DATA
1895                  <1>      ; 17/04/2021
1896 00001C6C FEC2          <1>      inc     dl ; dx = 3C5h
1897 00001C6E B003          <1>      mov     al, 3
1898 00001C70 EE           <1>      out     dx, al
1899 00001C71 B401          <1>      mov     ah, 1
1900                  <1>      _sm_8:
1901 00001C73 88E0          <1>      mov     al, ah
1902                  <1>      ;mov     dx, 3C4h ; VGAREG_SEQU_ADDRESS
1903                  <1>      ;dec     dx
1904                  <1>      ; 17/04/2021
1905 00001C75 FECA          <1>      dec     dl ; dx = 3C4h
1906 00001C77 EE           <1>      out     dx, al ; index
1907 00001C78 AC           <1>      lodsb
1908                  <1>      ;inc     dx ; 3C5h ; VGAREG_SEQU_DATA
1909                  <1>      ; 17/04/2021
1910 00001C79 FEC2          <1>      inc     dl
1911 00001C7B EE           <1>      out     dx, al

```

```

1912 00001C7C 80FC04    <1>    cmp     ah, 4 ; number of sequ regs
1913 00001C7F 7304    <1>    jnb     short _sm_9
1914 00001C81 FEC4    <1>    inc     ah
1915 00001C83 EBEE    <1>    jmp     short _sm_8
1916    <1>    _sm_9:
1917    <1>    ; Set GrafX Ctl
1918 00001C85 89DE    <1>    mov     esi, ebx ; addr of params tbl for selected mode
1919 00001C87 83C637    <1>    add     esi, 55 ; grdc regs
1920 00001C8A 30E4    <1>    xor     ah, ah ; 0
1921    <1>    _sm_10:
1922 00001C8C 88E0    <1>    mov     al, ah
1923    <1>    ;mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
1924    <1>    ; 17/04/2021
1925 00001C8E B2CE    <1>    mov     dl, 0CEh
1926 00001C90 EE    <1>    out     dx, al
1927 00001C91 AC    <1>    lodsb
1928    <1>    ;inc     dx ; 3CFh ; VGAREG_GRDC_DATA
1929    <1>    ; 17/04/2021
1930 00001C92 FEC2    <1>    inc     dl ; 3CFh
1931 00001C94 EE    <1>    out     dx, al
1932 00001C95 FEC4    <1>    inc     ah
1933 00001C97 80FC09    <1>    cmp     ah, 9 ; number of grdc regs
1934 00001C9A 72F0    <1>    jnb     short _sm_10
1935    <1>    ;
1936    <1>    ; Disable CRTC write protection
1937    <1>    ;mov     dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
1938    <1>    ; 17/04/2021
1939 00001C9C B2D4    <1>    mov     dl, 0D4h
1940    <1>    ;mov     al, 11h
1941    <1>    ;out     dx, al
1942    <1>    ;inc     dx
1943    <1>    ;sub     al, al
1944    <1>    ;out     dx, al
1945 00001C9E 66B81100    <1>    mov     ax, 11h
1946 00001CA2 66EF    <1>    out     dx, ax
1947 00001CA4 89DE    <1>    mov     esi, ebx ; addr of params tbl for selected mode
1948 00001CA6 83C60A    <1>    add     esi, 10 ; crtc regs
1949    <1>    ; ah = 0
1950    <1>    _sm_11:
1951 00001CA9 88E0    <1>    mov     al, ah
1952    <1>    ; dx = 3D4h = VGAREG_VGA_CRTC_ADDRESS
1953 00001CAB EE    <1>    out     dx, al ; index
1954 00001CAC AC    <1>    lodsb
1955    <1>    ;inc     dx ; VGAREG_VGA_CRTC_ADDRESS + 1
1956    <1>    ; 17/04/2021
1957 00001CAD FEC2    <1>    inc     dl
1958 00001CAF EE    <1>    out     dx, al ; value
1959 00001CB0 80FC18    <1>    cmp     ah, 24 ; number of crtc registers - 1
1960 00001CB3 7306    <1>    jnb     short _sm_12
1961 00001CB5 FEC4    <1>    inc     ah
1962    <1>    ;dec     dx ; 3D4h
1963    <1>    ; 17/04/2021
1964 00001CB7 FECA    <1>    dec     dl
1965 00001CB9 EBEE    <1>    jmp     short _sm_11
1966    <1>    _sm_12:
1967    <1>    ; Set the misc register
1968    <1>    ;mov     dx, 3CCh ; VGAREG_READ_MISC_OUTPUT
1969    <1>    ; 17/04/2021
1970 00001CBB B2CC    <1>    mov     dl, 0CCh
1971 00001CBD 8A4309    <1>    mov     al, [ebx+9] ; misc reg
1972 00001CC0 EE    <1>    out     dx, al
1973    <1>    ;
1974    <1>    ; Enable video
1975    <1>    ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
1976    <1>    ; 17/04/2021
1977 00001CC1 B2C0    <1>    mov     dl, 0C0h
1978 00001CC3 B020    <1>    mov     al, 20h
1979 00001CC5 EE    <1>    out     dx, al ; set bit 5 to 1
1980    <1>    ;mov     dx, 3DAh ; VGAREG_ACTL_RESET
1981    <1>    ; 17/04/2021
1982 00001CC6 B2DA    <1>    mov     dl, 0DAh
1983 00001CC8 EC    <1>    in      al, dx
1984    <1>    ;
1985    <1>    ; 17/11/2020
1986    <1>    ;cmp     byte [noclearmem], 0
1987    <1>    ;ja     short _sm_15
1988    <1>    ;
1989 00001CC9 F605[37830100]80    <1>    test    byte [noclearmem], 80h
1990 00001CD0 753B    <1>    jnz     short _sm_15
1991    <1>    ;
1992    <1>    ; 29/07/2016
1993 00001CD2 31C0    <1>    xor     eax, eax
1994    <1>    ;mov     ecx, 4000h ; 16K words (32K)
1995    <1>    ; 02/08/2022
1996 00001CD4 29C9    <1>    sub     ecx, ecx
1997 00001CD6 B540    <1>    mov     ch, 40h
1998    <1>    ; ecx = 4000h
1999 00001CD8 803D[4E830100]02    <1>    cmp     byte [VGA_MTYPE], 2 ; CTEXT, MTEXT, CGA
2000 00001CDF 7715    <1>    ja     short _sm_14 ; no ? (0A0000h)
2001 00001CE1 BF00800B00    <1>    mov     edi, 0B8000h
2002 00001CE6 7409    <1>    je     short _sm_13 ; CGA graphics mode
2003    <1>    ; 08/08/2016
2004 00001CE8 A3[4A830100]    <1>    mov     [VGA_INT43H], eax ; 0 ; default font
2005 00001CED 66B82007    <1>    mov     ax, 0720h ; CGA text mode
2006    <1>    _sm_13:
2007 00001CF1 F366AB    <1>    rep     stosw
2008 00001CF4 EB17    <1>    jmp     short _sm_15
2009    <1>    ;
2010    <1>    _sm_14:
2011 00001CF6 BF0000A00    <1>    mov     edi, 0A0000h
2012    <1>    ; ecx = 16384 dwords (64K)
2013    <1>    ;mov     dx, 3C4h ; VGAREG_SEQU_ADDRESS
2014    <1>    ; 17/04/2021
2015 00001CFB B2C4    <1>    mov     dl, 0C4h
2016 00001CFD B002    <1>    mov     al, 2
2017 00001CFF EE    <1>    out     dx, al
2018    <1>    ;mov     dx, 3C5h ; VGAREG_SEQU_DATA
2019    <1>    ;inc     dx
2020    <1>    ; 17/04/2021
2021 00001D00 FEC2    <1>    inc     dl ; 3C5h
2022 00001D02 EC    <1>    in      al, dx ; mmask
2023    <1>    ;push     ax
2024    <1>    ; 12/04/2021
2025 00001D03 50    <1>    push    eax
2026 00001D04 B00F    <1>    mov     al, 0Fh ; all planes
2027 00001D06 EE    <1>    out     dx, al
2028 00001D07 30C0    <1>    xor     al, al ; 0
2029 00001D09 F3AB    <1>    rep     stosd ; ecx = 163684 (64K)
2030    <1>    ;pop     ax
2031    <1>    ; 12/04/2021
2032 00001D0B 58    <1>    pop     eax
2033 00001D0C EE    <1>    out     dx, al ; mmask
2034    <1>    _sm_15:
2035    <1>    ; ebx = addr of params tbl for selected mode

```

```

2036      <1>      ; 10/08/2016
2037 00001D0D 668B03 <1>      mov     ax, [ebx] ; num of columns, 'twidth'
2038 00001D10 A2[20680000] <1>      mov     [CRT_COLS], al
2039      <1>      ;; 26/07/2016
2040      <1>      ;; CRTC_ADDRESS = 3D4h (always)
2041      <1>      ;mov     ah, [ebx+1] ; num of rows, 'theightm1'
2042 00001D15 FEC4 <1>      inc     ah ; 09/07/2016
2043 00001D17 8825[26680000] <1>      mov     [VGA_ROWS], ah
2044      <1>      ; 10/08/2016
2045 00001D1D 8A4302 <1>      mov     al, [ebx+2]
2046 00001D20 A2[22680000] <1>      mov     [CHAR_HEIGHT], al
2047      <1>      ; 29/07/2016
2048      <1>      ; length of regen buffer in bytes
2049 00001D25 668B4B03 <1>      mov     cx, [ebx+3] ; 'slength_1'
2050 00001D29 66890D[38830100] <1>      mov     [CRT_LEN], cx
2051      <1>      ;
2052      <1>      ; 27/07/2016
2053 00001D30 30E4 <1>      xor     ah, ah
2054 00001D32 A0[CE760100] <1>      mov     al, [ACTIVE_PAGE] ; may be > 0 for mode 3
2055      <1>      ;mul     word [CRT_LEN] ; 4096 for mode 3
2056 00001D37 66F7E1 <1>      mul     cx ; 29/07/2016
2057 00001D3A 66A3[BC760100] <1>      mov     [CRT_START], ax
2058      <1>      ;
2059 00001D40 B060 <1>      mov     al, 60h
2060      <1>      ;cmp     byte [noclearmem], 0
2061      <1>      ;jna     short _sm_16
2062      <1>      ;add     al, 80h
2063 00001D42 0A05[37830100] <1>      or      al, [noclearmem] ; 17/11/2020
2064      <1>      _sm_16:
2065 00001D48 A2[23680000] <1>      mov     [VGA_VIDEO_CTL], al
2066 00001D4D C605[24680000]F9 <1>      mov     byte [VGA_SWITCHES], 0F9h
2067 00001D54 8025[25680000]7F <1>      and     byte [VGA_MODESET_CTL], 7Fh
2068      <1>      ;
2069 00001D5B 5E <1>      pop     esi ; *
2070      <1>      ;
2071      <1>      ; 26/07/2016
2072      <1>      ; 07/07/2016
2073 00001D5C 668B0D[37680000] <1>      mov     cx, [CURSOR_MODE] ; restore cursor mode (initial value)
2074 00001D63 66870E <1>      xchg    cx, [esi] ; cl = start line, ch = end line
2075      <1>      ; reset to initial value
2076 00001D66 86CD <1>      xchg    ch, cl ; ch = start line, cl = end line
2077 00001D68 66890D[37680000] <1>      mov     [CURSOR_MODE], cx ; save (fixed) cursor mode
2078      <1>      ;
2079      <1>      ; 27/07/2016
2080 00001D6F 803D[4E830100]02 <1>      cmp     byte [VGA_MTYPE], 2 ; CTEXT, MTEXT
2081 00001D76 7317 <1>      jnb     short _sm_17
2082      <1>      ;
2083      <1>      ; Set cursor shape
2084      <1>      ;mov     cx, 0607h
2085      <1>      ;call    _set_ctype
2086      <1>      ;
2087      <1>      ; 29/07/2016
2088 00001D78 B40A <1>      mov     ah, 10 ; 6845 register for cursor set
2089 00001D7A E81F060000 <1>      call    m16 ; output cx register
2090      <1>      ;
2091      <1>      ; 25/07/2016
2092 00001D7F 803D[1E680000]03 <1>      cmp     byte [CRT_MODE], 03h
2093 00001D86 7507 <1>      jne     short _sm_17
2094      <1>      ; 26/07/2016
2095      <1>      ;
2096 00001D88 A0[CE760100] <1>      mov     al, [ACTIVE_PAGE]
2097 00001D8D EB0B <1>      jmp     short _sm_18
2098      <1>      _sm_17:
2099      <1>      ; Set cursor pos for page 0..7
2100      <1>      ;sub     ax, ax ; eax = 0
2101 00001D8F 29C0 <1>      sub     eax, eax ; 17/11/2020
2102 00001D91 BF[BE760100] <1>      mov     edi, CURSOR_POSN
2103 00001D96 AB <1>      stosd
2104 00001D97 AB <1>      stosd
2105 00001D98 AB <1>      stosd
2106 00001D99 AB <1>      stosd
2107      <1>      ;; Set active page 0
2108      <1>      ;mov     [ACTIVE_PAGE], al ; 0
2109      <1>      _sm_18:
2110      <1>      ; 29/07/2016
2111 00001D9A 803D[4E830100]02 <1>      cmp     byte [VGA_MTYPE], 2 ; CTEXT, MTEXT
2112      <1>      ;jnb     _sm_23
2113      <1>      ; 04/08/2022
2114 00001DA1 7205 <1>      jnb     short _sm_24
2115 00001DA3 E90C020000 <1>      jmp     _set_active_page
2116      <1>      _sm_24:
2117      <1>      ;cmp     byte [CHAR_HEIGHT], 16
2118      <1>      ;je      short _sm_19
2119      <1>      ;
2120      <1>      ;; copy and activate 8x16 font
2121      <1>      ;
2122      <1>      ; 26/07/2016
2123 00001DA8 B004 <1>      mov     al, 04h
2124      <1>      ;sub     bl, bl
2125      <1>      ; AX = 1104H ; Load ROM 8x16 Character Set
2126      <1>      ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
2127 00001DAA E844160000 <1>      call    load_text_8_16_pat
2128      <1>      ;
2129      <1>      ; video_func_1103h:
2130      <1>      ; biosfn_set_text_block_specifier:
2131      <1>      ; BL = font block selector code
2132      <1>      ; NOTE: TRDOS 386 only uses and sets font block 0
2133      <1>      ; (It is as BL = 0 for TRDOS 386)
2134 00001DAF 66BAC403 <1>      mov     dx, 3C4h ; VGAREG_SEQU_ADDRESS
2135      <1>      ;mov     ah, bl
2136      <1>      ;sub     ah, ah ; 0
2137      <1>      ;mov     al, 03h
2138      <1>      ; 19/11/2020
2139 00001DB3 66B80300 <1>      mov     ax, 03h
2140 00001DB7 66EF <1>      out     dx, ax
2141      <1>      _sm_19:
2142      <1>      ; 29/07/2016
2143      <1>      ; 26/07/2016
2144      <1>      ; 24/06/2016
2145      <1>      ;mov     edi, 0B8000h
2146      <1>      ;mov     cx, 4000h ; 16k words (32k)
2147      <1>      ;
2148 00001DB9 30C0 <1>      xor     al, al
2149 00001DBB 3805[35830100] <1>      cmp     [p_crt_mode], al ; 0
2150 00001DC1 7705 <1>      ja      short _sm_20 ; 03h, 80h or 83h
2151      <1>      ;
2152      <1>      ; case 1 - 19/11/2020
2153      <1>      ;
2154      <1>      ; If [pc_crt_mode] = 0, that means, previous mode is 03h
2155      <1>      ; and current mode is not 03h (case 1)
2156      <1>      ;
2157      <1>      ; 30/07/2016
2158      <1>      ; 24/06/2016
2159      <1>      ; TRDOS 386 (TRDOS v2) 'set mode' modification

```

```

2160      <1>      ; (for multiscreen feature):
2161      <1>      ; If '_set_mode' procedure is called for video mode 3
2162      <1>      ; while video mode is 3, video page will be cleared
2163      <1>      ; and cursor position of video page will be reset.
2164      <1>      ; If '_set_mode' procedure is called for a video mode
2165      <1>      ; except video mode 3, while current video mode
2166      <1>      ; is 3, all video pages of mode 3 will be copied
2167      <1>      ; to 98000h address as backup, before mode change.
2168      <1>      ; If '_set_mode' procedure is called for video mode 3
2169      <1>      ; while video mode is not 3 and if there is video
2170      <1>      ; page backup for video mode 3, all (of 8) mode 3
2171      <1>      ; video pages will be restored from 98000h.
2172      <1>
2173      <1>      ; 19/11/2020
2174      <1>      ; mov     [ACTIVE_PAGE], al ; 0
2175      <1>
2176      <1>      ; Here,
2177      <1>      ; video memory already cleared if [noclearmem] <> 80h
2178      <1>
2179      <1>      ; mov     ax, 0720h
2180      <1>      ; mov     cx, 4000h ; 16K words (32K)
2181      <1>      ; mov     edi, 0B8000h
2182      <1>      ; rep     stosw
2183      <1>      ; sub     al, al
2184      <1>
2185      <1>      ; jmp     short _sm_23
2186      <1>
2187      <1>      ; Set hardware side for the new active video page
2188      <1>
2189      00001DC3 E9EC010000      <1>      jmp     _set_active_page ; 19/11/2020
2190      <1>
2191      <1>      _sm_20:
2192      <1>      ; 19/11/2020
2193      <1>      ; case 2 or case 3 or case 4 - 19/11/2020
2194      <1>
2195      <1>      ; 19/11/2020
2196      00001DC8 803D[1E680000]03      <1>      cmp     byte [CRT_MODE], 3 ; new video mode
2197      00001DCF 754E      <1>      jne     short _sm_22 ; al = 0 (& video mode <> 03h)
2198      <1>      ; case 4 - 19/11/2020
2199      <1>      ; ([p_crt_mode] = 83h)
2200      <1>
2201      <1>      ; case 2 or case 3 - 19/11/2020
2202      <1>
2203      <1>      ; movzx   ebx, byte [ACTIVE_PAGE]
2204      <1>      ; 19/11/2020
2205      00001DD1 A0[36830100]      <1>      mov     al, [p_crt_page] ; previous mode 3 active page
2206      <1>      ; movzx   ebx, al
2207      <1>      ; shl     bl, 1 ; * 2
2208      <1>      ; add     ebx, CURSOR_POSN
2209      <1>
2210      <1>      ; 29/07/2016
2211      00001DD6 F605[35830100]7F      <1>      test    byte [p_crt_mode], 7Fh ; 83h or 80h or 03h
2212      00001DDD 740F      <1>      jz      short _sm_21 ; do not restore video pages
2213      <1>      ; case 2 - 19/11/2020
2214      <1>      ; case 3 - 19/11/2020
2215      <1>
2216      <1>      ; ([p_crt_mode] = 03h)
2217      <1>
2218      <1>      ; New video mode is 3 while current video mode is not 3
2219      <1>      ; (multi screen) video pages will be restored from 098000h
2220      <1>
2221      <1>      ; 19/11/2020
2222      00001DDF A2[CE760100]      <1>      mov     [ACTIVE_PAGE], al ; current mode 3 active page
2223      <1>
2224      <1>      ; 12/12/2020
2225      <1>      ; restore video pages
2226      <1>      ; mov     esi, 98000h ; 30/07/2016
2227      <1>      ; mov     edi, 0B8000h
2228      <1>      ; mov     cx, 2000h ; 8K dwords (32K)
2229      <1>      ; rep     movsd
2230      <1>
2231      <1>      ; 19/11/2020
2232      <1>      ; mov     [p_crt_mode], cl ; reset ('case 3' end condition)
2233      <1>
2234      <1>      ; restore cursor positions
2235      <1>      ; mov     esi, cursor_pposn
2236      <1>      ; mov     edi, CURSOR_POSN
2237      <1>      ; mov     ecx, 4 ; restore all cursor positions (16 bytes)
2238      <1>      ; mov     cl, 4
2239      <1>      ; rep     movsd
2240      <1>
2241      <1>      ; 12/12/2020
2242      00001DE4 E89A000000      <1>      call    _restore_mode3_multiscreen
2243      <1>
2244      <1>      ; jmp     short _sm_23 ; do not clear current video pages
2245      <1>
2246      <1>      ; 19/11/2020
2247      00001DE9 E9C6010000      <1>      jmp     _set_active_page
2248      <1>
2249      <1>      _sm_21:
2250      <1>      ; 19/11/2020
2251      <1>      ; case 2
2252      <1>
2253      <1>      ; ([p_crt_mode] = 80h)
2254      <1>
2255      <1>      ; User has requested to set video mode 3 again while
2256      <1>      ; current video mode is 3.. that means, set mode 03h
2257      <1>      ; parameters again and clear video page.
2258      <1>      ; ('noclearmem' option effects the result)
2259      <1>
2260      <1>      ; 19/11/2020
2261      00001DEE F605[37830100]80      <1>      test    byte [noclearmem], 80h
2262      00001DF5 7528      <1>      jnz     short _sm_22 ; 'do not clear video memory'
2263      <1>      ; continue with current text
2264      <1>      ; (user's option/choice)
2265      <1>
2266      <1>      ; clear video page
2267      <1>      ; mov     cx, [CRT_LEN] ; 4096
2268      00001DF7 66B82007      <1>      shr     cx, 1 ; 2048 ; 16/11/2020
2269      <1>      ; mov     ax, 0720h
2270      <1>      ; 26/11/2020
2271      <1>      ; mov     ecx, 2048 ; 4096/2
2272      <1>      ; 02/08/2022
2273      00001DFB 29C9      <1>      sub     ecx, ecx
2274      00001DFD B508      <1>      mov     ch, 08h
2275      <1>      ; ecx = 0800h
2276      00001DFF BF00800B00      <1>      mov     edi, 0B8000h ; [crt_base]
2277      00001E04 66033D[BC760100]      <1>      add     di, [CRT_START]
2278      00001E0B F366AB      <1>      rep     stosw ; FILL THE REGEN BUFFER WITH BLANKS
2279      <1>
2280      <1>      ; 19/11/2020
2281      00001E0E A0[CE760100]      <1>      mov     al, [ACTIVE_PAGE] ; 0 to 7 (for video mode 3)
2282      00001E13 0FB6D8      <1>      movzx   ebx, al
2283      00001E16 D0E3      <1>      shl     bl, 1
2283      00001E18 66898B[BE760100]      <1>      mov     [ebx+CURSOR_POSN], cx ; reset cursor position

```

```

2284 <1> _sm_22:
2285 <1> ;mov [p_crt_mode], al ; 0 ; reset
2286 <1> ; 19/11/2020
2287 <1> ;and byte [p_crt_mode], 3 ; 83h -> 3, 80h -> 0
2288 00001E1F 8025[35830100]7F <1> and byte [p_crt_mode], 7Fh ; 83h -> 3, 80h -> 0
2289 <1> _sm_23:
2290 <1> ; al = video page number
2291 <1> ; [CRT_LEN] = length of regen buffer in bytes
2292 <1> ;call _set_active_page
2293 <1> ; 16/11/2020
2294 00001E26 E989010000 <1> jmp _set_active_page
2295 <1>
2296 <1> ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
2297 <1> ;retn
2298 <1>
2299 <1> save_mode3_multiscreen:
2300 <1> ; 02/08/2022 (TRDOS v2.0.5)
2301 <1> ; 12/12/2020 (TRDOS v2.0.3)
2302 <1> ; save mode 03h video pages and cursor positions
2303 <1> ;
2304 <1> ; Modified registers: ecx (=0), esi, edi
2305 <1>
2306 <1> ; 12/12/2020
2307 <1> ; moved here from '_set_mode'
2308 <1> ; 03/07/2016
2309 <1> ; save video pages
2310 00001E2B BE00800B00 <1> mov esi, 0B8000h
2311 00001E30 BF00800900 <1> mov edi, 98000h ; 30/07/2016
2312 <1> ;mov ecx, (0B8000h-0B0000h)/4
2313 <1> ; 02/08/2022
2314 00001E35 29C9 <1> sub ecx, ecx
2315 00001E37 B520 <1> mov ch, 20h
2316 <1> ; ecx = 2000h
2317 00001E39 F3A5 <1> rep movsd
2318 <1>
2319 00001E3B C605[35830100]03 <1> mov byte [p_crt_mode], 3 ; previous mode, backup sign
2320 <1> ; 26/11/2020
2321 00001E42 860D[CE760100] <1> xchg cl, [ACTIVE_PAGE]
2322 00001E48 880D[36830100] <1> mov [p_crt_page], cl ; save as previous active page
2323 <1>
2324 <1> ; save cursor positions
2325 00001E4E BE[BE760100] <1> mov esi, CURSOR_POSN
2326 00001E53 BF[3A830100] <1> mov edi, cursor_pposn ; cursor positions backup
2327 00001E58 B104 <1> mov cl, 4
2328 00001E5A F3A5 <1> rep movsd
2329 00001E5C C3 <1> retn
2330 <1>
2331 <1> restore_mode3_multiscreen:
2332 <1> ; 02/08/2022 (TRDOS v2.0.5)
2333 <1> ; 12/12/2020 (TRDOS v2.0.3)
2334 <1> ; restore mode 03h video pages and cursor positions
2335 <1> ;
2336 <1> ; Input:
2337 <1> ; settings from the last 'save_mode3_multiscreen'
2338 <1> ;
2339 <1> ; Output:
2340 <1> ; AL = active video page = [ACTIVE_PAGE]
2341 <1> ;
2342 <1> ; Modified registers: al, ecx (=0), esi, edi
2343 <1>
2344 00001E5D A0[36830100] <1> mov al, [p_crt_page] ; previous mode 3 active page
2345 00001E62 A2[CE760100] <1> mov [ACTIVE_PAGE], al ; current mode 3 active page
2346 <1>
2347 <1> ; 12/12/2020
2348 <1> ; moved here from 'vesa_vbe3_pmi'
2349 <1>
2350 <1> ; 07/12/2020
2351 <1> ; restore CRT_START according to ACTIVE_PAGE
2352 <1> ;mov [CRT_START], cx ; 0
2353 <1> ; 12/12/2020
2354 00001E67 66C705[BC760100]00- <1> mov word [CRT_START], 0
2355 00001E6F 00 <1>
2356 <1>
2357 00001E70 08C0 <1> ; check active page and set it again if it is not 0
2358 <1> or al, al
2359 <1> ;jz short vbe3_pmi_7
2360 00001E72 740F <1> ;jz short _restore_mode3_multiscreen
2361 00001E74 88C1 <1> jz short r_m3_ms_1
2362 <1> mov cl, al
2363 <1> ;vbe3_pmi_5:
2364 00001E76 668105[BC760100]00- <1> r_m3_ms_0:
2365 00001E7E 10 <1> add word [CRT_START], 4096
2366 00001E7F FEC9 <1> dec cl
2367 00001E81 75F3 <1> ;jnz short vbe3_pmi_5
2368 <1> jnz short r_m3_ms_0
2369 <1> r_m3_ms_1:
2370 <1> ; 12/12/2020
2371 <1> ; moved here from '_set_mode'
2372 <1> _restore_mode3_multiscreen:
2373 <1> ; Modified registers: ecx, esi, edi
2374 <1>
2375 00001E83 BE00800900 <1> ; restore video pages
2376 00001E88 BF00800B00 <1> mov esi, 98000h ; 30/07/2016
2377 <1> mov edi, 0B8000h
2378 <1> ;mov cx, 2000h ; 8K dwords (32K)
2379 <1> ;mov ecx, 2000h
2380 00001E8D 29C9 <1> ; 02/08/2022
2381 00001E8F B520 <1> sub ecx, ecx
2382 <1> mov ch, 20h
2383 00001E91 F3A5 <1> ; ecx = 2000h
2384 <1> rep movsd
2385 <1>
2386 00001E93 880D[35830100] <1> ; 19/11/2020
2387 <1> mov [p_crt_mode], cl ; reset ('case 3' end condition)
2388 <1>
2389 00001E99 BE[3A830100] <1> ; restore cursor positions
2390 00001E9E BF[BE760100] <1> mov esi, cursor_pposn
2391 <1> mov edi, CURSOR_POSN
2392 00001EA3 B104 <1> ;mov ecx, 4 ; restore all cursor positions (16 bytes)
2393 00001EA5 F3A5 <1> mov cl, 4
2394 00001EA7 C3 <1> rep movsd
2395 <1> retn
2396 <1>
2397 <1> cursor_shape_fix:
2398 <1> ; 12/04/2021
2399 <1> ; 07/07/2016
2400 <1> ; (Cursor start and cursor end line values -6,7-
2401 <1> ; will be fixed depending on character height)
2402 <1> ;
2403 <1> ; derived from 'Plex86/Bochs VGABios' source code
2404 <1> ; vgabios-0.7a (2011)
2405 <1> ; by the LGPL VGABios developers Team (2001-2008)
2406 <1> ; 'vgabios.c', 'biosfn_set_cursor_shape (CH,CL)'

```

```

2406 <1> ;
2407 <1> ; INPUT ->
2408 <1> ; AL = cursor start line (=6)
2409 <1> ; AH = cursor end line (=7)
2410 <1> ; OUTPUT ->
2411 <1> ; AL = cursor start line (=14)
2412 <1> ; AH = cursor end line (=15)
2413 <1> ;
2414 <1> ;; if((modeset_ctl&0x01)&&(cheight>8)&&(CL<8)&&(CH<0x20))
2415 <1>
2416 <1> ;test byte [VGA_MODESET_CTL], 1 ; VGA active
2417 <1> ;jz short csf_3
2418 00001EA8 803D[22680000]08 <1> cmp byte [CHAR_HEIGHT], 8
2419 00001EAF 7647 <1> jna short csf_3
2420 00001EB1 80FC08 <1> cmp ah, 8
2421 00001EB4 7342 <1> jnb short csf_3
2422 00001EB6 3C20 <1> cmp al, 20h
2423 00001EB8 733E <1> jnb short csf_3
2424 <1> ;
2425 <1> ;push ax
2426 <1> ; 12/04/2021
2427 00001EBA 50 <1> push eax
2428 <1> ; {
2429 <1> ; if(CL!=(CH+1))
2430 00001EBB FEC0 <1> inc al
2431 00001EBD 38C4 <1> cmp ah, al ; ah != al + 1
2432 00001EBF 740F <1> je short csf_1
2433 <1> ; CH = ((CH+1) * cheight / 8) - 1;
2434 00001EC1 8A25[22680000] <1> mov ah, [CHAR_HEIGHT]
2435 00001EC7 F6E4 <1> mul ah
2436 00001EC9 C0E803 <1> shr al, 3 ; / 8
2437 00001ECC FEC8 <1> dec al ; - 1
2438 00001ECE EB0E <1> jmp short csf_2
2439 <1> csf_1:
2440 <1> ; }
2441 <1> ; else ; ah = al + 1
2442 <1> ; {
2443 00001ED0 FEC4 <1> inc ah ; ah = ah + 1
2444 <1> ; CH = ((CL+1) * cheight / 8) - 2;
2445 00001ED2 A0[22680000] <1> mov al, [CHAR_HEIGHT]
2446 00001ED7 F6E4 <1> mul ah
2447 00001ED9 C0E803 <1> shr al, 3 ; / 8
2448 00001EDC 2C02 <1> sub al, 2 ; - 2
2449 <1> ; al = 14 (if [CHAR_HEIGHT] = 16)
2450 <1> csf_2:
2451 00001EDE 880424 <1> mov [esp], al
2452 00001EE1 8A642401 <1> mov ah, [esp+1]
2453 <1> ; CL = ((CL+1) * cheight / 8) - 1;
2454 00001EE5 FEC4 <1> inc ah
2455 00001EE7 A0[22680000] <1> mov al, [CHAR_HEIGHT]
2456 00001EEC F6E4 <1> mul ah
2457 00001EEE C0E803 <1> shr al, 3 ; / 8
2458 00001EF1 FEC8 <1> dec al ; - 1
2459 00001EF3 88442401 <1> mov [esp+1], al
2460 <1> ; ah = 15 (if [CHAR_HEIGHT] = 16)
2461 <1> ;
2462 <1> ;pop ax
2463 <1> ; 12/04/2021
2464 00001EF7 58 <1> pop eax
2465 <1> csf_3:
2466 00001EF8 C3 <1> retn
2467 <1>
2468 <1> SET_CTYPE:
2469 <1> ; 04/08/2022 (TRDOS 386 v2.0.5)
2470 <1> ; 12/09/2016
2471 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
2472 00001EF9 803D[1E680000]07 <1> cmp byte [CRT_MODE], 7
2473 <1> ;ja VIDEO_RETURN ; 12/09/2016
2474 <1> ; 04/08/2022
2475 00001F00 7738 <1> ja short set_cpos_inv_vp
2476 00001F02 E805000000 <1> call _set_ctype
2477 00001F07 E96CFCFFFF <1> jmp VIDEO_RETURN
2478 <1>
2479 <1> _set_ctype:
2480 <1> ; 02/09/2014 (Retro UNIX 386 v1)
2481 <1> ;
2482 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
2483 <1> ;
2484 <1> ; (CH) = BITS 4-0 = START LINE FOR CURSOR
2485 <1> ; ** HARDWARE WILL ALWAYS CAUSE BLINK
2486 <1> ; ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC BLINKING
2487 <1> ; OR NO CURSOR AT ALL
2488 <1> ; (CL) = BITS 4-0 = END LINE FOR CURSOR
2489 <1> ;
2490 <1> ;-----
2491 <1> ; SET_CTYPE
2492 <1> ; THIS ROUTINE SETS THE CURSOR VALUE
2493 <1> ; INPUT
2494 <1> ; (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
2495 <1> ; OUTPUT
2496 <1> ; NONE
2497 <1> ;-----
2498 <1>
2499 <1> ; 02/08/2022 (TRDOS 386 kernel v2.0.5)
2500 <1> ;
2501 <1> ; 07/07/2016
2502 <1> ; Fixing cursor start and stop line depending on
2503 <1> ; current character height (=16)
2504 <1> ; (Note: Default/initial values are 6 and 7.
2505 <1> ; If set values are 6 (start) & 7 (stop) and
2506 <1> ; [CHAR_HEIGHT] = 16 :
2507 <1> ; After fixing, start line will be 14, stop line
2508 <1> ; will be 15.)
2509 <1> ;
2510 <1> ;mov ax, cx
2511 <1> ; 02/08/2022
2512 00001F0C 89C8 <1> mov eax, ecx
2513 <1> ;
2514 00001F0E 86E0 <1> xchg al, ah
2515 <1> ; AL = start line, AH = stop line
2516 00001F10 E893FFFFFF <1> call cursor_shape_fix
2517 <1> ; AL = start line (fixed), AH = stop line (fixed)
2518 <1> ;mov cx, ax
2519 <1> ; 02/08/2022
2520 00001F15 89C1 <1> mov ecx, eax
2521 00001F17 86CD <1> xchg ch, cl
2522 <1> ; CH = start line (fixed), CL = stop line (fixed)
2523 <1> ;
2524 00001F19 B40A <1> mov ah, 10 ; 6845 register for cursor set
2525 00001F1B 66890D[37680000] <1> mov [CURSOR_MODE], cx ; save in data area
2526 <1> ;call m16 ; output cx register
2527 <1> ;retn
2528 00001F22 E977040000 <1> jmp m16
2529 <1>

```

```

2530      <1> SET_CPOS:
2531      <1> ; 02/08/2022 (TRDOS 386 kernel v2.0.5)
2532      <1> ; 12/09/2016
2533      <1> ; 07/07/2016
2534      <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
2535 00001F27 80FF07      <1> cmp     bh, 7 ; video page > 7 ; 07/07/2016
2536      <1> ;ja     VIDEO_RETURN
2537      <1> ; 02/08/2022
2538 00001F2A 770E      <1> ja      short set_cpos_inv_vp
2539      <1> ;
2540 00001F2C 803D[1E680000]07      <1> cmp     byte [CRT_MODE], 7
2541 00001F33 770A      <1> ja      short vga_set_cpos ; 12/09/2016
2542 00001F35 E839040000      <1> call    _set_cpos
2543      <1> set_cpos_inv_vp: ; 02/08/2022
2544 00001F3A E939FCFFFF      <1> jmp     VIDEO_RETURN
2545      <1>
2546      <1> vga_set_cpos:
2547      <1> ; 12/09/2016
2548      <1> ; 09/07/2016
2549      <1> ; set cursor position
2550      <1> ; NOTE: Hardware cursor position will not be set
2551      <1> ; in any VGA modes (>7)
2552      <1> ; But, cursor position will be saved into
2553      <1> ; [CURSOR_POSN].
2554      <1> ; TRDOS 386 (TRDOS v2.0) uses only one page
2555      <1> ; (page 0) for all graphics modes.
2556      <1>
2557 00001F3F 668915[BE760100]      <1> mov     [CURSOR_POSN], dx ; save cursor pos for pg 0
2558      <1> ; 04/08/2016
2559      <1> ;mov    bh, [ACTIVE_PAGE] ; = 0
2560      <1> ;call   _set_cpos
2561 00001F46 E92DFCFFFF      <1> jmp     VIDEO_RETURN
2562      <1>
2563      <1> READ_CURSOR:
2564      <1> ; 12/09/2016
2565      <1> ; 07/07/2016
2566      <1> ; 12/05/2016
2567      <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
2568      <1> ;
2569      <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
2570      <1>
2571      <1> ;-----
2572      <1> ; READ_CURSOR
2573      <1> ; THIS ROUTINE READS THE CURRENT CURSOR VALUE FROM THE
2574      <1> ; 845, FORMATS IT, AND SENDS IT BACK TO THE CALLER
2575      <1> ; INPUT
2576      <1> ; BH - PAGE OF CURSOR
2577      <1> ; OUTPUT
2578      <1> ; DX - ROW, COLUMN OF THE CURRENT CURSOR POSITION
2579      <1> ; CX - CURRENT CURSOR MODE
2580      <1> ;-----
2581      <1>
2582      <1> ; BH = video page number (0 to 7)
2583      <1>
2584      <1> ; 07/07/2016
2585 00001F4B 80FF07      <1> cmp     bh, 7 ; video page > 7 (invalid)
2586 00001F4E 7606      <1> jna     short read_cursor_1
2587      <1> ; invalid video page (input)
2588 00001F50 31C9      <1> xor     ecx, ecx ; 0
2589 00001F52 31D2      <1> xor     edx, edx ; 0
2590 00001F54 EB15      <1> jmp     short read_cursor_2
2591      <1> read_cursor_1:
2592      <1> ; 12/09/2016
2593 00001F56 803D[1E680000]07      <1> cmp     byte [CRT_MODE], 7 ; vga mode
2594 00001F5D 7727      <1> ja      short vga_get_cpos
2595      <1> ;
2596 00001F5F E815000000      <1> call    get_cpos
2597 00001F64 0FB70D[37680000]      <1> movzx   ecx, word [CURSOR_MODE]
2598      <1> read_cursor_2:
2599      <1> pop     ebp
2600 00001F6C 5F      <1> pop     edi
2601 00001F6D 5E      <1> pop     esi
2602 00001F6E 5B      <1> pop     ebx
2603 00001F6F 58      <1> pop     eax ; DISCARD SAVED CX AND DX
2604 00001F70 58      <1> pop     eax
2605 00001F71 A1[28830100]      <1> mov     eax, [video_eax] ; 12/05/2017
2606      <1> ;;15/01/2017
2607      <1> ;;mov    byte [intflg], 0 ; 07/01/2017
2608 00001F76 1F      <1> pop     ds
2609 00001F77 07      <1> pop     es
2610 00001F78 CF      <1> iretd
2611      <1>
2612      <1> get_cpos:
2613      <1> ; 12/05/2016
2614      <1> ; 16/01/2016
2615      <1> ; BH = video page number (0 to 7)
2616      <1> ;
2617 00001F79 D0E7      <1> shl     bh, 1 ; WORD OFFSET
2618 00001F7B 0FB6F7      <1> movzx   esi, bh
2619 00001F7E 0FB796[BE760100]      <1> movzx   edx, word [esi+CURSOR_POSN]
2620 00001F85 C3      <1> retn
2621      <1>
2622      <1> vga_get_cpos:
2623      <1> ; 12/09/2016
2624      <1> ; get cursor position (vga)
2625 00001F86 0FB715[BE760100]      <1> movzx   edx, word [CURSOR_POSN] ; cursor pos for pg 0
2626 00001F8D 31C9      <1> xor     ecx, ecx ; Cursor Mode = 0 (invalid)
2627 00001F8F EBDA      <1> jmp     short read_cursor_2
2628      <1>
2629      <1> ACT_DISP_PAGE:
2630      <1> ; 07/07/2016
2631      <1> ; 26/06/2016
2632      <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
2633      <1> ;
2634      <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
2635      <1> ;
2636      <1> ;-----
2637      <1> ; ACT_DISP_PAGE
2638      <1> ; THIS ROUTINE SETS THE ACTIVE DISPLAY PAGE, ALLOWING
2639      <1> ; THE FULL USE OF THE MEMORY SET ASIDE FOR THE VIDEO ATTACHMENT
2640      <1> ; INPUT
2641      <1> ; AL HAS THE NEW ACTIVE DISPLAY PAGE
2642      <1> ; OUTPUT
2643      <1> ; THE 6845 IS RESET TO DISPLAY THAT PAGE
2644      <1> ;-----
2645      <1> ; 07/07/2016
2646 00001F91 3C07      <1> cmp     al, 7 ; > 7 = invalid video page number
2647      <1> ;ja     VIDEO_RETURN
2648 00001F93 7715      <1> ja      short adp_2 ; 18/11/2020
2649      <1> ;cmp     byte [CRT_MODE], 3
2650      <1> ;je      short adp_1
2651      <1> ; 18/11/2020
2652 00001F95 8A25[1E680000]      <1> mov     ah, [CRT_MODE]
2653 00001F9B 80FC03      <1> cmp     ah, 3

```

```

2654 00001F9E 7605      <1>      jna      short adp_1 ; mode 01h, 00h (01h), 02h (03h), 03h
2655 00001FA0 80FC07    <1>      cmp      ah, 7      ; mode 07h (03h)
2656 00001FA3 7505      <1>      jne      short adp_2
2657                      <1>      ;and      al, al
2658                      <1>      ;jnz      VIDEO_RETURN
2659                      <1>      ;;sub    al, al ; 0 ; force to page 0
2660                      <1>      adp_1:
2661 00001FA5 E805000000  <1>      call     set_active_page
2662                      <1>      adp_2:
2663 00001FAA E9C9FBFFFF  <1>      jmp      VIDEO_RETURN
2664                      <1>
2665                      <1>      set_active_page:      ; tty_sw
2666                      <1>      ; 02/08/2022 (TRDOS 386 kernel v2.0.5)
2667                      <1>      ; 09/12/2017
2668                      <1>      ; 26/07/2016
2669                      <1>      ; 26/06/2016
2670                      <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
2671                      <1>      ; 30/06/2015
2672                      <1>      ; 04/03/2014 (act_disp_page --> tty_sw)
2673                      <1>      ; 10/12/2013
2674                      <1>      ; 04/12/2013
2675                      <1>      ;
2676 00001FAF A2[CE760100] <1>      mov      [ACTIVE_PAGE], al ; save active page value ; [ptty]
2677                      <1>      _set_active_page:
2678                      <1>      ; 27/06/2015
2679                      <1>      ;movzx  ebx, al
2680                      <1>      ; 02/08/2022
2681 00001FB4 0FB6C0      <1>      movzx    eax, al
2682 00001FB7 89C3        <1>      mov      ebx, eax
2683                      <1>      ;
2684                      <1>      ;cbw      ; 07/09/2014 (ah=0)
2685                      <1>      ; 02/08/2022
2686                      <1>      ;sub     ah, ah ; 09/12/2017
2687 00001FB9 66F725[38830100] <1>      mul      word [CRT_LEN] ; get saved length of regen buffer
2688                      <1>      ; display page times regen length
2689                      <1>      ; 10/12/2013
2690 00001FC0 66A3[BC760100] <1>      mov      [CRT_START], ax ; save start address for later
2691                      <1>      ;mov     cx, ax ; start address to cx
2692                      <1>      ; 02/08/2022
2693 00001FC6 89C1        <1>      mov      ecx, eax
2694                      <1>      _M16:
2695                      <1>      ;;sar     cx, 1
2696                      <1>      ;shr     cx, 1 ; divide by 2 for 6845 handling
2697                      <1>      ; 02/08/2022
2698 00001FC8 D1E9        <1>      shr      ecx, 1
2699 00001FCA B40C        <1>      mov      ah, 12 ; 6845 register for start address
2700 00001FCC E8CD030000  <1>      call     m16
2701                      <1>      ;sal     bx, 1
2702                      <1>      ; 01/09/2014
2703 00001FD1 D0E3        <1>      shl      bl, 1 ; *2 for word offset
2704 00001FD3 81C3[BE760100] <1>      add      ebx, CURSOR_POSN
2705 00001FD9 668B13      <1>      mov      dx, [ebx] ; get cursor for this page
2706                      <1>      ; 16/01/2016
2707                      <1>      ;call    m18
2708                      <1>      ;retn
2709 00001FDC E9A9030000  <1>      jmp      m18
2710                      <1>
2711                      <1>      position:
2712                      <1>      ; 02/08/2022 - TRDOS 386 v2.0.5
2713                      <1>      ; 17/04/2021 - TRDOS 386 v2.0.4
2714                      <1>      ; 24/06/2016
2715                      <1>      ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
2716                      <1>      ; 27/06/2015
2717                      <1>      ; 02/09/2014
2718                      <1>      ; 30/08/2014 (Retro UNIX 386 v1)
2719                      <1>      ; 04/12/2013 (Retro UNIX 8086 v1)
2720                      <1>      ;
2721                      <1>      ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
2722                      <1>      ;
2723                      <1>      ;-----
2724                      <1>      ; POSITION
2725                      <1>      ; THIS SERVICE ROUTINE CALCULATES THE REGEN BUFFER ADDRESS
2726                      <1>      ; OF A CHARACTER IN THE ALPHA MODE
2727                      <1>      ; INPUT
2728                      <1>      ; AX = ROW, COLUMN POSITION
2729                      <1>      ; OUTPUT
2730                      <1>      ; AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
2731                      <1>      ;-----
2732                      <1>
2733                      <1>      ; DX = ROW, COLUMN POSITION
2734                      <1>      ;movzx  eax, byte [CRT_COLS] ; 27/06/2015
2735 00001FE1 31C0        <1>      xor      eax, eax ; 02/09/2014
2736 00001FE3 B050        <1>      mov      al, 80 ; determine bytes to row
2737 00001FE5 F6E6        <1>      mul      dh ; row value
2738                      <1>      ;xor     dh, dh ; 0
2739                      <1>      ;add     ax, dx ; add column value to the result
2740                      <1>      ; 16/04/2021
2741 00001FE7 00D0        <1>      add      al, dl
2742 00001FE9 80D400      <1>      adc      ah, 0
2743                      <1>      ; 02/08/2022
2744 00001FEC D1E0        <1>      shl      eax, 1
2745                      <1>      ;shl     ax, 1 ; * 2 for attribute bytes
2746                      <1>      ; EAX = AX = OFFSET OF CHAR POSITION IN REGEN BUFFER
2747 00001FEE C3          <1>      retn
2748                      <1>
2749                      <1>      find_position:
2750                      <1>      ; 17/04/2021 - TRDOS 386 v2.0.4
2751                      <1>      ; 24/06/2016
2752                      <1>      ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
2753                      <1>      ; 27/06/2015
2754                      <1>      ; 07/09/2014
2755                      <1>      ; 02/09/2014
2756                      <1>      ; 30/08/2014 (Retro UNIX 386 v1)
2757                      <1>      ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
2758                      <1>
2759 00001FEF 0FB6CF      <1>      movzx    ecx, bh ; video page number
2760                      <1>      ; 17/04/2021
2761                      <1>      ;mov     esi, ecx
2762                      <1>      ;shl     si, 1
2763                      <1>      ;mov     dx, [esi+CURSOR_POSN]
2764                      <1>      ;jz      short p21
2765                      <1>      ;xor     si, si
2766                      <1>      ; 17/04/2021
2767 00001FF2 31F6        <1>      xor      esi, esi
2768 00001FF4 D0E1        <1>      shl      cl, 1
2769 00001FF6 668B91[BE760100] <1>      mov      dx, [ecx+CURSOR_POSN]
2770 00001FFD 740B        <1>      jz      short p21
2771 00001FFF D0E9        <1>      shr      cl, 1
2772                      <1>      p20:
2773 00002001 660335[38830100] <1>      add      si, [CRT_LEN] ; 24/06/2016
2774                      <1>      ;add     si, 80*25*2 ; add length of buffer for one page
2775 00002008 E2F7        <1>      loop     p20
2776                      <1>      p21:
2777 0000200A 6621D2      <1>      and      dx, dx

```



```

2778 0000200D 7407      <1>      jz      short p22
2779 0000200F E8CDFFFFFF <1>      call   position ; determine location in regen in page
2780 00002014 01C6      <1>      add    esi, eax ; add location to start of regen page
2781      <1>      p22:
2782      <1>      ;mov    dx, [addr_6845] ; get base address of active display
2783      <1>      ;mov    dx, 03D4h ; I/O address of color card
2784      <1>      ;add    dx, 6 ; point at status port
2785 00002016 66BADA03   <1>      mov    dx, 03DAh ; status port
2786      <1>      ; cx = 0
2787 0000201A C3         <1>      retn
2788      <1>
2789      <1>      SCROLL_UP:
2790      <1>      ; 04/08/2022 (TRDOS 386 v2.0.5)
2791      <1>      ; 07/07/2016
2792      <1>      ; 26/06/2016
2793      <1>      ; 12/05/2016
2794      <1>      ; 30/01/2016
2795      <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
2796      <1>      ; 07/09/2014
2797      <1>      ; 02/09/2014
2798      <1>      ; 01/09/2014 (Retro UNIX 386 v1 - beginning)
2799      <1>      ; 04/04/2014
2800      <1>      ; 04/12/2013
2801      <1>
2802      <1>      ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
2803      <1>
2804      <1>      -----
2805      <1>      SCROLL UP
2806      <1>      THIS ROUTINE MOVES A BLOCK OF CHARACTERS UP
2807      <1>      ON THE SCREEN
2808      <1>      INPUT
2809      <1>      (AH) = CURRENT CRT MODE
2810      <1>      (AL) = NUMBER OF ROWS TO SCROLL
2811      <1>      (CX) = ROW/COLUMN OF UPPER LEFT CORNER
2812      <1>      (DX) = ROW/COLUMN OF LOWER RIGHT CORNER
2813      <1>      (BH) = ATTRIBUTE TO BE USED ON BLANKED LINE
2814      <1>      (DS) = DATA SEGMENT
2815      <1>      (ES) = REGEN BUFFER SEGMENT
2816      <1>      OUTPUT
2817      <1>      NONE -- THE REGEN BUFFER IS MODIFIED
2818      <1>      -----
2819      <1>
2820      <1>      ; 07/07/2016
2821 0000201B 38F5      <1>      cmp     ch, dh
2822      <1>      ;ja     VIDEO_RETURN
2823      <1>      ; 04/08/2022
2824 0000201D 7709      <1>      ja     short _s_u_retn
2825      <1>
2826 0000201F 38D1      <1>      cmp     cl, dl
2827      <1>      ;ja     VIDEO_RETURN
2828      <1>      ; 04/08/2022
2829 00002021 7705      <1>      ja     short _s_u_retn
2830      <1>
2831 00002023 E805000000 <1>      call    _scroll_up
2832      <1>      _s_u_retn:
2833 00002028 E94BF8FFFF <1>      jmp     VIDEO_RETURN
2834      <1>
2835      <1>      _scroll_up: ; from 'write_tty'
2836      <1>      ;
2837      <1>      ; cl = left upper column
2838      <1>      ; ch = left upper row
2839      <1>      ; dl = right lower column
2840      <1>      ; dh = right lower row
2841      <1>      ;
2842      <1>      ; al = line count
2843      <1>      ; bl = attribute to be used on blanked line
2844      <1>      ; bh = video page number (0 to 7)
2845      <1>
2846 0000202D E89B000000 <1>      call    test_line_count ; 16/01/2016
2847      <1>
2848 00002032 8A25[1E680000] <1>      mov     ah, [CRT_MODE] ; current video mode
2849      <1>      ;cmp    byte [CRT_MODE], 4
2850      <1>      ;cmp    ah, 4 ; 07/07/2016
2851      <1>      ;jnb    GRAPHICS_UP ; 26/06/2016
2852      <1>      ; 18/11/2020
2853 00002038 80FC04      <1>      cmp     ah, 4
2854 0000203B 720A      <1>      jb     short n0
2855 0000203D 80FC07      <1>      cmp     ah, 7 ; TEST FOR BW CARD
2856      <1>      ; (80x25 text, mono)
2857 00002040 7405      <1>      je     short n0 ; same with mode 3 for TRDOS 386
2858 00002042 E91B050000 <1>      jmp     GRAPHICS_UP
2859      <1>      n0:
2860      <1>      ; 07/07/2016
2861 00002047 80FF07      <1>      cmp     bh, 7 ; video page number
2862 0000204A 7606      <1>      jna     short n1
2863 0000204C 8A3D[CE760100] <1>      mov     bh, [ACTIVE_PAGE]
2864      <1>      n1:
2865 00002052 88DC      <1>      mov     ah, bl ; attribute
2866      <1>      ;push    ax ; *
2867      <1>      ; 12/04/2021
2868 00002054 50         <1>      push    eax ; *
2869      <1>      ;mov     esi, [CRT_BASE]
2870 00002055 BE00800B00 <1>      mov     esi, 0B8000h
2871 0000205A 3A3D[CE760100] <1>      cmp     bh, [ACTIVE_PAGE]
2872 00002060 750B      <1>      jne     short n2
2873      <1>      ;
2874 00002062 66A1[BC760100] <1>      mov     ax, [CRT_START]
2875 00002068 6601C6      <1>      add     si, ax
2876 0000206B EB11      <1>      jmp     short n4
2877      <1>      n2:
2878 0000206D 20FF      <1>      and     bh, bh
2879 0000206F 740D      <1>      jz     short n4
2880 00002071 88F8      <1>      mov     al, bh
2881      <1>      n3:
2882 00002073 660335[38830100] <1>      add     si, [CRT_LEN]
2883 0000207A FEC8      <1>      dec     al
2884 0000207C 75F5      <1>      jnz     short n3
2885      <1>      n4:
2886 0000207E E85B000000 <1>      call    scroll_position ; 16/01/2016
2887 00002083 7420      <1>      jz     short n6
2888      <1>
2889 00002085 01CE      <1>      add     esi, ecx ; from address for scroll
2890 00002087 88F5      <1>      mov     ch, dh ; #rows in block
2891 00002089 28C5      <1>      sub     ch, al ; #rows to be moved
2892      <1>      n5:
2893 0000208B E88A000000 <1>      call    n10 ; 16/01/2016
2894      <1>
2895 00002090 51         <1>      push    ecx
2896 00002091 0FB60D[20680000] <1>      movzx    ecx, byte [CRT_COLS]
2897 00002098 00C9      <1>      add     cl, cl
2898 0000209A 01CE      <1>      add     esi, ecx ; next line
2899 0000209C 01CF      <1>      add     edi, ecx
2900 0000209E 59         <1>      pop     ecx
2901      <1>

```

```

2902 0000209F FECD      <1>      dec     ch      ; count of lines to move
2903 000020A1 75E8      <1>      jnz     short n5 ; row loop
2904                                <1>      ; ch = 0
2905 000020A3 88C6      <1>      mov     dh, al   ; #rows
2906                                <1> n6:
2907                                <1>      ; attribute in ah
2908 000020A5 B020      <1>      mov     al, ' ' ; fill with blanks
2909                                <1> n7:
2910 000020A7 E87B000000 <1>      call    n11 ; 16/01/2016
2911                                <1>
2912 000020AC 8A0D[20680000] <1>      mov     cl, [CRT_COLS]
2913 000020B2 00C9      <1>      add     cl, cl
2914 000020B4 01CF      <1>      add     edi, ecx
2915                                <1>
2916 000020B6 FECE      <1>      dec     dh
2917 000020B8 75ED      <1>      jnz     short n7
2918                                <1> n16:
2919 000020BA 3A3D[CE760100] <1>      cmp     bh, [ACTIVE_PAGE]
2920 000020C0 750A      <1>      jne     short n8
2921                                <1>
2922                                <1>      ; cmp     byte [CRT_MODE], 7 ; is this the black and white card
2923                                <1>      ; je      short n8 ; if so, skip the mode reset
2924                                <1>
2925 000020C2 A0[1F680000] <1>      mov     al, [CRT_MODE_SET] ; get the value of mode set
2926 000020C7 66BAD803 <1>      mov     dx, 03D8h ; always set color card port
2927 000020CB EE         <1>      out     dx, al
2928                                <1> n8:
2929 000020CC C3         <1>      retn
2930                                <1>
2931                                <1> test_line_count:
2932                                <1>      ; 12/04/2021
2933                                <1>      ; 12/05/2016
2934                                <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
2935                                <1>      ; 07/09/2014 (scroll_up)
2936 000020CD 08C0      <1>      or      al, al
2937 000020CF 740C      <1>      jz      short al_set2
2938                                <1>      ; push    dx
2939                                <1>      ; 12/04/2021
2940 000020D1 52         <1>      push    edx
2941 000020D2 28EE      <1>      sub     dh, ch ; subtract upper row from lower row number
2942 000020D4 FEC6      <1>      inc     dh ; adjust difference by 1
2943 000020D6 38C6      <1>      cmp     dh, al ; line count = amount of rows in window?
2944 000020D8 7502      <1>      jne     short al_set1 ; if not the we're all set
2945 000020DA 30C0      <1>      xor     al, al ; otherwise set al to zero
2946                                <1> al_set1:
2947                                <1>      ; pop     dx
2948                                <1>      ; 12/04/2021
2949 000020DC 5A         <1>      pop     edx
2950                                <1> al_set2:
2951 000020DD C3         <1>      retn
2952                                <1>
2953                                <1> scroll_position:
2954                                <1>      ; 12/04/2021
2955                                <1>      ; 26/06/2016
2956                                <1>      ; 30/01/2016
2957                                <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
2958                                <1>      ; 07/09/2014 (scroll_up)
2959                                <1>
2960                                <1>      ; (*) [esp+4] = ax (al = line count, ah = attribute)
2961                                <1>
2962                                <1>      ; push    dx
2963                                <1>      ; 12/04/2021
2964 000020DE 52         <1>      push    edx
2965 000020DF 6689CA      <1>      mov     dx, cx ; now, upper left position in DX
2966 000020E2 E8FAFEFFFF <1>      call    position
2967 000020E7 01C6      <1>      add     esi, eax
2968 000020E9 89F7      <1>      mov     edi, esi
2969                                <1>      ; pop     dx ; lower right position in DX
2970                                <1>      ; 12/04/2021
2971 000020EB 5A         <1>      pop     edx
2972 000020EC 6629CA      <1>      sub     dx, cx
2973 000020EF FEC6      <1>      inc     dh ; dh = #rows
2974 000020F1 FEC2      <1>      inc     dl ; dl = #cols in block
2975 000020F3 59         <1>      pop     ecx ; return address
2976                                <1>      ; pop     ax ; * ; al = line count, ah = attribute
2977                                <1>      ; 12/04/2021
2978 000020F4 58         <1>      pop     eax ; (*)
2979 000020F5 51         <1>      push    ecx ; return address
2980 000020F6 0FB7C8      <1>      movzx   ecx, ax
2981 000020F9 8A25[20680000] <1>      mov     ah, [CRT_COLS]
2982 000020FF F6E4      <1>      mul     ah ; determine offset to from address
2983                                <1>      ; add     ax, ax ; *2 for attribute byte
2984                                <1>      ; 02/08/2022
2985                                <1>      ; shl     eax, 1
2986 00002101 01C0      <1>      add     eax, eax
2987                                <1>
2988                                <1>      ; push    ax ; offset
2989                                <1>      ; push    dx
2990                                <1>      ; 12/04/2021
2991 00002103 50         <1>      push    eax ; offset
2992 00002104 52         <1>      push    edx
2993                                <1>
2994                                <1>      ; 04/04/2014
2995 00002105 66BADA03 <1>      mov     dx, 3DAh ; guaranteed to be color card here
2996                                <1> n9:
2997 00002109 EC         <1>      in      al, dx ; wait_display_enable
2998 0000210A A808      <1>      test    al, RVRT ; wait for vertical retrace
2999 0000210C 74FB      <1>      jz      short n9 ; wait_display_enable
3000 0000210E B025      <1>      mov     al, 25h
3001 00002110 B2D8      <1>      mov     dl, 0D8h ; address control port
3002 00002112 EE         <1>      out     dx, al ; turn off video during vertical retrace
3003                                <1>      ; pop     dx ; #rows, #cols
3004                                <1>      ; pop     ax ; offset
3005                                <1>      ; 12/04/2021
3006 00002113 5A         <1>      pop     edx ; #rows, #cols
3007 00002114 58         <1>      pop     eax ; offset
3008 00002115 6691      <1>      xchg    ax, cx ;
3009                                <1>      ; ecx = offset, al = line count, ah = attribute
3010                                <1>
3011 00002117 08C0      <1>      or      al, al
3012 00002119 C3         <1>      retn
3013                                <1> n10:
3014                                <1>      ; Move rows
3015 0000211A 88D1      <1>      mov     cl, dl ; get # of cols to move
3016 0000211C 56         <1>      push    esi
3017 0000211D 57         <1>      push    edi ; save start address
3018                                <1> n10r:
3019 0000211E 66A5      <1>      movsw   ; move that line on screen
3020 00002120 FEC9      <1>      dec     cl
3021 00002122 75FA      <1>      jnz     short n10r
3022 00002124 5F         <1>      pop     edi
3023 00002125 5E         <1>      pop     esi ; recover addresses
3024 00002126 C3         <1>      retn
3025                                <1> n11:

```

```

3026      <1>      ; clear rows
3027      <1>      ; dh = #rows
3028      00002127 88D1      <1>      mov cl, dl ; get # of cols to clear
3029      00002129 57      <1>      push edi ; save address
3030      <1>      n11r:
3031      0000212A 66AB      <1>      stosw ; store fill character
3032      0000212C FEC9      <1>      dec cl
3033      0000212E 75FA      <1>      jnz short n11r
3034      00002130 5F      <1>      pop edi ; recover address
3035      00002131 C3      <1>      retn
3036      <1>
3037      <1>      SCROLL_DOWN:
3038      <1>      ; 12/04/2021
3039      <1>      ; 07/07/2016
3040      <1>      ; 27/06/2016
3041      <1>      ; 26/06/2016
3042      <1>      ; 12/05/2016
3043      <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
3044      <1>      ;
3045      <1>      ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
3046      <1>
3047      <1>      ;-----
3048      <1>      SCROLL DOWN
3049      <1>      ; THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
3050      <1>      ; BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
3051      <1>      ; WITH A DEFINED CHARACTER
3052      <1>      ; INPUT
3053      <1>      ; (AH) = CURRENT CRT MODE
3054      <1>      ; (AL) = NUMBER OF LINES TO SCROLL
3055      <1>      ; (CX) = UPPER LEFT CORNER OF REGION
3056      <1>      ; (DX) = LOWER RIGHT CORNER OF REGION
3057      <1>      ; (BH) = FILL CHARACTER
3058      <1>      ; (DS) = DATA SEGMENT
3059      <1>      ; (ES) = REGEN SEGMENT
3060      <1>      ; OUTPUT
3061      <1>      ; NONE -- SCREEN IS SCROLLED
3062      <1>      ;-----
3063      <1>
3064      <1>      ; 07/07/2016
3065      00002132 38F5      <1>      cmp ch, dh
3066      <1>      ;ja VIDEO_RETURN
3067      00002134 7709      <1>      ja short _s_d_retn ; 18/11/2020
3068      00002136 38D1      <1>      cmp cl, dl
3069      <1>      ;ja VIDEO_RETURN
3070      00002138 7705      <1>      ja short _s_d_retn ; 18/11/2020
3071      <1>      ;
3072      0000213A E805000000 <1>      call _scroll_down
3073      <1>      _s_d_retn:
3074      0000213F E934FAFFFF <1>      jmp VIDEO_RETURN
3075      <1>
3076      <1>      _scroll_down: ; 27/06/2016
3077      <1>
3078      <1>      ; cl = left upper column
3079      <1>      ; ch = left upper row
3080      <1>      ; dl = right lower column
3081      <1>      ; dh = right lower row
3082      <1>      ;
3083      <1>      ; al = line count
3084      <1>      ; bl = attribute to be used on blanked line
3085      <1>      ; bh = video page number (0 to 7)
3086      <1>
3087      <1>      ; !!!!
3088      00002144 FD      <1>      std ; DIRECTION FOR SCROLL DOWN
3089      <1>      ; !!!!
3090      00002145 E883FFFFFF <1>      call test_line_count ; 16/01/2016
3091      <1>
3092      0000214A 8A25[1E680000] <1>      mov ah, [CRT_MODE] ; current video mode
3093      <1>      ;cmp byte [CRT_MODE], 4
3094      <1>      ;cmp ah, 4 ; 07/07/2016
3095      <1>      ;jnb GRAPHICS_DOWN ; 26/06/2016
3096      <1>      ; 18/11/2020
3097      00002150 80FC04      <1>      cmp ah, 4
3098      00002153 720A      <1>      jb short _n0
3099      00002155 80FC07      <1>      cmp ah, 7 ; TEST FOR BW CARD
3100      <1>      ; (80x25 text, mono)
3101      00002158 7405      <1>      je short _n0 ; same with mode 3 for TRDOS 386
3102      0000215A E9D5060000 <1>      jmp GRAPHICS_DOWN
3103      <1>      _n0:
3104      <1>      ; 07/07/2016
3105      0000215F 80FF07      <1>      cmp bh, 7 ; video page number
3106      00002162 7606      <1>      jna short n12
3107      00002164 8A3D[CE760100] <1>      mov bh, [ACTIVE_PAGE]
3108      <1>      ;
3109      <1>      n12: ; CONTINUE_DOWN
3110      0000216A 88DC      <1>      mov ah, bl
3111      <1>      ;push ax ; * ; save attribute in ah
3112      <1>      ; 12/04/2021
3113      0000216C 50      <1>      push eax
3114      0000216D 6689D0      <1>      mov ax, dx ; LOWER RIGHT CORNER
3115      00002170 E869FFFFFF <1>      call scroll_position ; GET REGEN LOCATION
3116      00002175 741F      <1>      jz short n14
3117      00002177 29CE      <1>      sub esi, ecx ; SI IS FROM ADDRESS
3118      00002179 88F5      <1>      mov ch, dh ; #rows in block
3119      0000217B 28C5      <1>      sub ch, al ; #rows to be moved
3120      <1>      n13:
3121      0000217D E898FFFFFF <1>      call n10 ; MOVE ONE ROW
3122      <1>
3123      00002182 51      <1>      push ecx
3124      00002183 8A0D[20680000] <1>      mov cl, [CRT_COLS]
3125      00002189 00C9      <1>      add cl, cl
3126      0000218B 29CE      <1>      sub esi, ecx ; next line
3127      0000218D 29CF      <1>      sub edi, ecx
3128      0000218F 59      <1>      pop ecx
3129      <1>
3130      00002190 FECD      <1>      dec ch ; count of lines to move
3131      00002192 75E9      <1>      jnz short n13 ; row loop
3132      <1>      ; ch = 0
3133      00002194 88C6      <1>      mov dh, al ; #rows
3134      <1>      n14:
3135      <1>      ; attribute in ah
3136      00002196 B020      <1>      mov al, ' ' ; fill with blanks
3137      <1>      n15:
3138      00002198 E88AFFFFFF <1>      call n11 ; 16/01/2016
3139      <1>
3140      0000219D 8A0D[20680000] <1>      mov cl, [CRT_COLS]
3141      000021A3 00C9      <1>      add cl, cl
3142      000021A5 29CF      <1>      sub edi, ecx
3143      <1>
3144      000021A7 FECE      <1>      dec dh
3145      000021A9 75ED      <1>      jnz short n15
3146      <1>      ;
3147      <1>      ; 18/11/2020
3148      000021AB FC      <1>      cld ; clear direction flag
3149      <1>      ;

```

```

3150 000021AC E909FFFFFF      <1>      jmp      n16 ; 27/06/2016
3151                          <1>
3152                          <1>      cmp      bh, [ACTIVE_PAGE]
3153                          <1>      jne      short n16
3154                          <1>
3155                          <1>      ;cmp      byte [CRT_MODE], 7      ; is this the black and white card
3156                          <1>      ;je      short n16                ; if so, skip the mode reset
3157                          <1>
3158                          <1>      mov      al, [CRT_MODE_SET] ; get the value of mode set
3159                          <1>      mov      dx, 03D8h ; always set color card port
3160                          <1>      out      dx, al
3161                          <1>      ;n16:
3162                          <1>      ; !!!!
3163                          <1>      cld                      ; Clear direction flag !
3164                          <1>      ; !!!!
3165                          <1>      retn
3166                          <1>
3167                          <1>      READ_AC_CURRENT:
3168                          <1>      ; 08/07/2016
3169                          <1>      ; 26/06/2016
3170                          <1>      ; 12/05/2016
3171                          <1>      ; 18/01/2016
3172                          <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
3173                          <1>
3174                          <1>      ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
3175                          <1>
3176                          <1>      ; 08/07/2016
3177 000021B1 803D[1E680000]07 <1>      cmp      byte [CRT_MODE], 7 ; 6!?
3178 000021B8 7607          <1>      jna      short read_ac_c
3179 000021BA 31C0          <1>      xor      eax, eax
3180 000021BC E9BCF9FFFF      <1>      jmp      _video_return
3181                          <1>      read_ac_c:
3182 000021C1 E805000000      <1>      call     _read_ac_current
3183                          <1>      ; 12/05/2016
3184                          <1>      ; jmp      VIDEO_RETURN
3185 000021C6 E9B2F9FFFF      <1>      jmp      _video_return
3186                          <1>
3187                          <1>      ;-----
3188                          <1>      READ_AC_CURRENT      :
3189                          <1>      THIS ROUTINE READS THE ATTRIBUTE AND CHARACTER AT THE CURRENT :
3190                          <1>      CURSOR POSITION AND RETURNS THEM TO THE CALLER      :
3191                          <1>      INPUT      :
3192                          <1>      (AH) = CURRENT CRT MODE      :
3193                          <1>      (BH) = DISPLAY PAGE ( ALPHA MODES ONLY )      :
3194                          <1>      (DS) = DATA SEGMENT      :
3195                          <1>      (ES) = REGEN SEGMENT      :
3196                          <1>      OUTPUT      :
3197                          <1>      (AL) = CHARACTER READ      :
3198                          <1>      (AH) = ATTRIBUTE READ      :
3199                          <1>      ;-----
3200                          <1>
3201                          <1>      _read_ac_current:
3202                          <1>      ; 26/06/2016
3203                          <1>      ; 12/05/2016
3204                          <1>      ; 18/01/2016
3205                          <1>
3206 000021CB 8A25[1E680000] <1>      mov      ah, [CRT_MODE] ; current video mode
3207 000021D1 80FC04          <1>      cmp      ah, 4
3208 000021D4 720A          <1>      jb      short p10
3209                          <1>      ; 18/11/2020
3210                          <1>      ; cmp      byte [CRT_MODE], 4
3211                          <1>      ; jnb      GRAPHICS_READ ; 26/06/2016
3212                          <1>
3213 000021D6 80FC07          <1>      cmp      ah, 7 ; TEST FOR BW CARD (80x25 monochrome text)
3214 000021D9 7405          <1>      je      short p10 ; same with mode 3 in TRDOS 386
3215 000021DB E9A0080000      <1>      jmp      GRAPHICS_READ
3216                          <1>      p10:
3217 000021E0 E80AFEFFFF      <1>      call     find_position ; GET REGEN LOCATION AND PORT ADDRESS
3218                          <1>
3219                          <1>      ; esi = regen location
3220                          <1>      ; dx = status port
3221                          <1>
3222                          <1>
3223                          <1>      ; 18/11/2020
3224                          <1>      ; convert display mode to a zero value
3225                          <1>      ; for 80 column color mode
3226                          <1>      ; mov      ah, [CRT_MODE]
3227                          <1>      ; sub      ah, 2
3228                          <1>      ; shr      ah, 1
3229                          <1>      ; jnz      short p13
3230                          <1>
3231                          <1>      ; 05/12/2020
3232                          <1>      ; 18/11/2020 (TRDOS 386 v2.0.3)
3233                          <1>      ; xor      bl, bl ; 0
3234 000021E5 803D[1E680000]03 <1>      cmp      byte [CRT_MODE], 03h ; 80x25 color text
3235                          <1>      ; je      short p11 ; Note: Only mode 03h and mode 01h are
3236                          <1>      ; inc      bl ; 1 ; in use by TRDOS 386 as text modes
3237                          <1>      ; jmp      short p14 ; (07h, 00h and 02h are redirected)
3238 000021EC 7516          <1>      jne      short p13
3239                          <1>
3240                          <1>      ; 05/12/2020
3241 000021EE 3A3D[CE760100] <1>      cmp      bh, [ACTIVE_PAGE]
3242 000021F4 750E          <1>      jne      short p13
3243                          <1>
3244                          <1>      ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
3245                          <1>      p11:
3246 000021F6 FB          <1>      sti                      ; enable interrupts first
3247                          <1>      ; 05/12/2020
3248 000021F7 90          <1>      nop
3249                          <1>      ; 18/11/2020
3250                          <1>      ; or      bl, bl
3251                          <1>      ; jnz      short p13 ; mode 01h (and mode 00h) - 40x25 color text
3252 000021F8 FA          <1>      cli                      ; block interrupts for single loop
3253 000021F9 EC          <1>      in      al, dx ; get status from the adapter
3254 000021FA A801          <1>      test     al, RHRZ ; is horizontal retrace low
3255 000021FC 75F8          <1>      jnz      short p11 ; wait until it is
3256                          <1>      p12:
3257 000021FE EC          <1>      in      al, dx ; get status again
3258 000021FF A809          <1>      test     al, RVRT+RHRZ ; is horizontal or vertical retrace high
3259 00002201 74FB          <1>      jz      short p12 ; wait until either retrace active
3260                          <1>      ;p14:
3261 00002203 FB          <1>      sti
3262                          <1>      p13:
3263 00002204 81C600800B00 <1>      add      esi, 0B8000h
3264 0000220A 668B06          <1>      mov      ax, [esi]
3265                          <1>
3266 0000220D C3          <1>      retn      ; 18/01/2016
3267                          <1>
3268                          <1>      WRITE_AC_CURRENT:
3269                          <1>      ; 08/07/2016
3270                          <1>      ; 26/06/2016
3271                          <1>      ; 24/06/2016
3272                          <1>      ; 12/05/2016
3273                          <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)

```

```

3274 <1> ;
3275 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
3276 <1> ;
3277 <1> -----
3278 <1> WRITE_AC_CURRENT :
3279 <1> THIS ROUTINE WRITES THE ATTRIBUTE AND CHARACTER :
3280 <1> AT THE CURRENT CURSOR POSITION :
3281 <1> INPUT :
3282 <1> (AH) = CURRENT CRT MODE :
3283 <1> (BH) = DISPLAY PAGE :
3284 <1> (CX) = COUNT OF CHARACTERS TO WRITE :
3285 <1> (AL) = CHAR TO WRITE :
3286 <1> (BL) = ATTRIBUTE OF CHAR TO WRITE :
3287 <1> (DS) = DATA SEGMENT :
3288 <1> (ES) = REGEN SEGMENT :
3289 <1> OUTPUT :
3290 <1> DISPLAY REGEN BUFFER UPDATED :
3291 <1> -----
3292 <1>
3293 <1> ; 08/07/2016
3294 0000220E 803D[1E680000]07 <1> cmp byte [CRT_MODE], 7 ; 6!?
3295 00002215 760A <1> jna short write_ac_c
3296 <1>
3297 00002217 E8B60A0000 <1> call vga_write_char_attr
3298 0000221C E957F9FFFF <1> jmp VIDEO_RETURN
3299 <1>
3300 <1> write_ac_c:
3301 00002221 E834000000 <1> call _write_c_current
3302 <1>
3303 00002226 0FB6F7 <1> movzx esi, bh ; video page number (0 to 7)
3304 00002229 889E[27680000] <1> mov [esi+chr_attr], bl ; color/attribute
3305 <1>
3306 0000222F E944F9FFFF <1> jmp VIDEO_RETURN
3307 <1>
3308 <1> WRITE_C_CURRENT:
3309 <1> ; 08/07/2016
3310 <1> ; 26/06/2016
3311 <1> ; 12/05/2016
3312 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
3313 <1>
3314 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
3315 <1> ;
3316 <1> -----
3317 <1> WRITE_C_CURRENT :
3318 <1> THIS ROUTINE WRITES THE CHARACTER AT :
3319 <1> THE CURRENT CURSOR POSITION, ATTRIBUTE UNCHANGED :
3320 <1> INPUT :
3321 <1> (AH) = CURRENT CRT MODE :
3322 <1> (BH) = DISPLAY PAGE :
3323 <1> (CX) = COUNT OF CHARACTERS TO WRITE :
3324 <1> (AL) = CHAR TO WRITE :
3325 <1> (DS) = DATA SEGMENT :
3326 <1> (ES) = REGEN SEGMENT :
3327 <1> OUTPUT :
3328 <1> DISPLAY REGEN BUFFER UPDATED :
3329 <1> -----
3330 <1>
3331 <1> ; 08/07/2016
3332 00002234 803D[1E680000]07 <1> cmp byte [CRT_MODE], 7 ; 6!?
3333 0000223B 760A <1> jna short write_c_c
3334 <1>
3335 0000223D E8900A0000 <1> call vga_write_char_only
3336 00002242 E931F9FFFF <1> jmp VIDEO_RETURN
3337 <1>
3338 <1> write_c_c:
3339 <1> ;and bh, 7 ; video page number (<= 7)
3340 00002247 0FB6F7 <1> movzx esi, bh
3341 0000224A 8A9E[27680000] <1> mov bl, [esi+chr_attr]
3342 <1>
3343 00002250 E805000000 <1> call _write_c_current
3344 00002255 E91EF9FFFF <1> jmp VIDEO_RETURN
3345 <1>
3346 <1> _write_c_current: ; from 'write_tty'
3347 <1> ; 12/04/2021
3348 <1> ; 18/11/2020
3349 <1> ; 26/06/2016
3350 <1> ; 24/06/2016
3351 <1> ; 12/05/2016
3352 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
3353 <1> ; 30/08/2014 (Retro UNIX 386 v1)
3354 <1> ; 18/01/2014
3355 <1> ; 04/12/2013
3356 <1> ;
3357 <1> ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
3358 <1> ;
3359 <1> ; 18/11/2020
3360 0000225A 8A25[1E680000] <1> mov ah, [CRT_MODE] ; current video mode
3361 00002260 80FC04 <1> cmp ah, 4
3362 00002263 720A <1> jb short p40
3363 <1>
3364 <1> ;cmp byte [CRT_MODE], 4
3365 <1> ;jnb GRAPHICS_WRITE ; 26/06/2016
3366 <1>
3367 00002265 80FC07 <1> cmp ah, 7 ; TEST FOR BW CARD
3368 00002268 7405 <1> je short p40
3369 0000226A E963070000 <1> jmp GRAPHICS_WRITE
3370 <1> p40:
3371 <1> ; al = character
3372 <1> ; bl = color/attribute
3373 <1> ; bh = video page
3374 <1> ; cx = count of characters to write
3375 <1> ;push dx
3376 <1> ; 12/04/2021
3377 0000226F 52 <1> push edx ; *
3378 00002270 88DC <1> mov ah, bl ; color/attribute (12/05/2016)
3379 <1> ;push ax ; save character & attribute/color
3380 <1> ;push cx
3381 <1> ; 12/04/2021
3382 00002272 50 <1> push eax ; ** ; save character & attribute/color
3383 00002273 51 <1> push ecx ; ***
3384 00002274 E876FDFFFF <1> call find_position ; get regen location and port address
3385 <1> ;pop cx
3386 <1> ; 12/04/2021
3387 00002279 59 <1> pop ecx ; ***
3388 <1> ; esi = regen location
3389 <1> ; dx = status port
3390 <1> ;
3391 0000227A 81C600800B00 <1> add esi, 0B8000h ; 30/08/2014 (crt_base)
3392 <1> ;
3393 <1> ; 18/11/2020
3394 <1> ; convert display mode to a zero value
3395 <1> ; for 80 column color mode
3396 <1> ;mov ah, [CRT_MODE]
3397 <1> ;sub ah, 2

```

```

3398      <1>      ;shr    ah, 1
3399      <1>      ;jnz    short p44      ; 26/06/2016
3400      <1>
3401      <1>      ; 05/12/2020
3402      <1>      ; 18/11/2020 (TRDOS 386 v2.0.3)
3403      <1>      ;xor     b1, b1 ; 0
3404 00002280 803D[1E680000]03 <1>      cmp     byte [CRT_MODE], 03h ; 80x25 color text
3405      <1>      ;je      short p41      ; Note: Only mode 03h and mode 01h are
3406      <1>      ;inc     b1      ; 1 ; in use by TRDOS 386 as text modes
3407      <1>      ;jmp     short p43      ; (07h, 00h and 02h are redirected)
3408      <1>      ; 05/12/2020
3409 00002287 751A <1>      jne     short p44
3410      <1>      p46:
3411      <1>      ; 05/12/2020
3412 00002289 3A3D[CE760100] <1>      cmp     bh, [ACTIVE_PAGE]
3413 0000228F 7512 <1>      jne     short p44
3414      <1>
3415      <1>      ; WAIT FOR HORIZONTAL RETRACE OR VERTICAL RETRACE IF COLOR 80
3416      <1>      p41:
3417      <1>      ; 05/12/2020
3418 00002291 FB <1>      sti     ; enable interrupts first
3419 00002292 90 <1>      nop
3420      <1>      ; 18/11/2020
3421      <1>      ;or      b1, b1
3422      <1>      ;jnz     short p44 ; mode 01h (and mode 00h) - 40x25 color text
3423 00002293 FA <1>      cli     ; block interrupts for single loop
3424 00002294 EC <1>      in      al, dx ; get status from the adapter
3425 00002295 A808 <1>      test    al, RVRT ; check for vertical retrace first
3426 00002297 7509 <1>      jnz     short p43 ; Do fast write now if vertical retrace
3427 00002299 A801 <1>      test    al, RHRZ ; is horizontal retrace low
3428 0000229B 75F4 <1>      jnz     short p41 ; wait until it is
3429      <1>      p42:
3430 0000229D EC <1>      in      al, dx ; get status again
3431 0000229E A809 <1>      test    al, RVRT+RHRZ ; is horizontal or vertical retrace high
3432 000022A0 74FB <1>      jz      short p42 ; wait until either retrace active
3433      <1>      p43:
3434 000022A2 FB <1>      sti
3435      <1>      p44:
3436 000022A3 668B0424 <1>      mov     ax, [esp] ; restore the character (al) & attribute (ah)
3437 000022A7 668906 <1>      mov     [esi], ax
3438      <1>
3439 000022AA 6649 <1>      dec     cx
3440 000022AC 740D <1>      jz      short p45
3441      <1>
3442 000022AE 46 <1>      inc     esi
3443 000022AF 46 <1>      inc     esi
3444      <1>
3445      <1>      ; 05/12/2020
3446 000022B0 803D[1E680000]03 <1>      cmp     byte [CRT_MODE], 03h
3447 000022B7 75EA <1>      jne     short p44
3448      <1>      ;jmp     short p41
3449 000022B9 EBCE <1>      jmp     short p46
3450      <1>      p45:
3451      <1>      ;pop     ax
3452      <1>      ;pop     dx
3453      <1>      ; 12/04/2021
3454 000022BB 58 <1>      pop     eax ; **
3455 000022BC 5A <1>      pop     edx ; *
3456 000022BD C3 <1>      retn
3457      <1>
3458      <1>      ; 09/07/2016
3459      <1>      ; 26/06/2016
3460      <1>      ; 24/06/2016
3461      <1>      ; 12/05/2016
3462      <1>      ; 18/01/2016
3463      <1>      ; 16/01/2016 - TRDOS 386 (TRDOS v2.0)
3464      <1>      ; 30/06/2015
3465      <1>      ; 27/06/2015
3466      <1>      ; 11/03/2015
3467      <1>      ; 02/09/2014
3468      <1>      ; 30/08/2014
3469      <1>      ; VIDEO FUNCTIONS
3470      <1>      ; (write_tty - Retro UNIX 8086 v1 - U9.ASM, 01/02/2014)
3471      <1>
3472      <1>      WRITE_TTY:
3473      <1>      ; 09/12/2017
3474      <1>      ; 09/07/2016
3475      <1>      ; 01/07/2016
3476      <1>      ; 26/06/2016
3477      <1>      ; 24/06/2016
3478      <1>      ; 13/05/2016
3479      <1>      ; 12/05/2016
3480      <1>      ; 30/01/2016
3481      <1>      ; 18/01/2016
3482      <1>      ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
3483      <1>      ; 13/08/2015
3484      <1>      ; 02/09/2014
3485      <1>      ; 30/08/2014 (Retro UNIX 386 v1 - beginning)
3486      <1>      ; 01/02/2014 (Retro UNIX 8086 v1 - last update)
3487      <1>      ; 03/12/2013 (Retro UNIX 8086 v1 - beginning)
3488      <1>      ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
3489      <1>      ;
3490      <1>      ; INPUT -> AL = Character to be written
3491      <1>      ; BL = Color (Forecolor, Backcolor)
3492      <1>      ; BH = Video Page (0 to 7)
3493      <1>
3494      <1>      ; 09/07/2016
3495 000022BE 803D[1E680000]07 <1>      cmp     byte [CRT_MODE], 7
3496 000022C5 760A <1>      jna     short write_tty_cga
3497      <1>
3498 000022C7 E8EB0C0000 <1>      call    vga_write_teletype
3499 000022CC E9A7F8FFFF <1>      jmp     VIDEO_RETURN
3500      <1>
3501      <1>      write_tty_cga:
3502      <1>      ; 13/05/2016
3503      <1>      ; call    _write_tty
3504      <1>      ; 01/07/2016
3505 000022D1 E818000000 <1>      call    _write_tty_m3
3506 000022D6 E99DF8FFFF <1>      jmp     VIDEO_RETURN
3507      <1>
3508      <1>      RVRTequ 00001000b ; VIDEO VERTICAL RETRACE BIT
3509      <1>      RHRZequ 00000001b ; VIDEO HORIZONTAL RETRACE BIT
3510      <1>
3511      <1>      ; Derived from "WRITE_TTY" procedure of IBM "pc-at" rombios source code
3512      <1>      ; (06/10/1985), 'video.asm', INT 10H, VIDEO_IO
3513      <1>
3514      <1>      ; 06/10/85 VIDEO DISPLAY BIOS
3515      <1>
3516      <1>      ;--- WRITE_TTY -----
3517      <1>      ;
3518      <1>      ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE :
3519      <1>      ; VIDEO CARDS. THE INPUT CHARACTER IS WRITTEN TO THE CURRENT :
3520      <1>      ; CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION. :
3521      <1>      ; IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN :

```

```

3522 <1> ; IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED. IF THE ROW :
3523 <1> ; ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST ROW, :
3524 <1> ; FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE LINE. :
3525 <1> ; WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING THE :
3526 <1> ; NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE PREVIOUS :
3527 <1> ; LINE BEFORE THE SCROLL, IN CHARACTER MODE. IN GRAPHICS MODE, :
3528 <1> ; THE 0 COLOR IS USED. :
3529 <1> ; ENTRY -- :
3530 <1> ; (AH) = CURRENT CRT MODE :
3531 <1> ; (AL) = CHARACTER TO BE WRITTEN :
3532 <1> ; NOTE THAT BACK SPACE, CARRIAGE RETURN, BELL AND LINE FEED ARE :
3533 <1> ; HANDLED AS COMMANDS RATHER THAN AS DISPLAY GRAPHICS CHARACTERS :
3534 <1> ; (BL) = FOREGROUND COLOR FOR CHAR WRITE IF CURRENTLY IN A GRAPHICS MODE :
3535 <1> ; EXIT -- :
3536 <1> ; ALL REGISTERS SAVED :
3537 <1> ;-----
3538 <1>
3539 <1> ; 02/08/2022 (TRDOS 386 v2.0.5)
3540 <1> ; 18/11/2020 (TRDOS 386 v2.0.3)
3541 <1> ; 09/12/2017
3542 <1> ; 08/07/2016
3543 <1> ; 26/06/2016
3544 <1> ; 24/06/2016
3545 <1> _write_tty:
3546 <1> ; 13/05/2016
3547 <1> ; --- 18/11/2020 ---
3548 <1> ; NOTE:
3549 <1> ; Only kernel calls "_write_tty" procedure...
3550 <1> ; TRDOS 386 v2 kernel uses video mode 3 for displaying
3551 <1> ; (some error) messages and also mainprog command interpreter
3552 <1> ; (in kernel) uses "_write_tty".
3553 <1> ; So, here video mode must be set to 3 if it is not 3.
3554 <1>
3555 000022DB FA <1> cli ; disable interrupts
3556 <1>
3557 <1> ; 01/09/2014
3558 000022DC 803D[1E680000]03 <1> cmp byte [CRT_MODE], 3
3559 000022E3 7409 <1> je short _write_tty_m3
3560 <1>
3561 <1> set_mode_3:
3562 000022E5 53 <1> push ebx
3563 000022E6 50 <1> push eax
3564 <1> ; call _set_mode
3565 <1> ; 18/11/2020
3566 000022E7 E89BF8FFFF <1> call set_txt_mode ; set video mode to 03h
3567 000022EC 58 <1> pop eax
3568 000022ED 5B <1> pop ebx
3569 <1>
3570 <1> _write_tty_m3: ; 24/06/2016 (m3 -> _write_tty_m3)
3571 000022EE 0FB6F7 <1> movzx esi, bh ; 12/05/2016
3572 <1> ; shl si, 1
3573 <1> ; 02/08/2022
3574 000022F1 D1E6 <1> shl esi, 1
3575 000022F3 81C6[BE760100] <1> add esi, CURSOR_POSN
3576 000022F9 668B16 <1> mov dx, [esi]
3577 <1>
3578 <1> ; dx now has the current cursor position
3579 <1>
3580 000022FC 3C0D <1> cmp al, 0Dh; CR ; is it carriage return or control character
3581 <1> ; jbe short u8
3582 <1> ; 17/04/2021
3583 000022FE 7241 <1> jb short u8
3584 00002300 746F <1> je short u9
3585 <1>
3586 <1> ; write the char to the screen
3587 <1> u0:
3588 <1> ; al = character
3589 <1> ; bl = attribute/color
3590 <1> ; bh = video page number (0 to 7)
3591 <1>
3592 <1> ; mov cx, 1 ; 24/06/2016
3593 <1> ; 02/08/2022
3594 00002302 29C9 <1> sub ecx, ecx
3595 00002304 FEC1 <1> inc cl ; ecx = 1
3596 <1> ; cx = count of characters to write
3597 <1>
3598 00002306 E84FFFFFFF <1> call _write_c_current ; 16/01/2015
3599 <1>
3600 <1> ; position the cursor for next char
3601 0000230B FEC2 <1> inc dl ; next column
3602 0000230D 3A15[20680000] <1> cmp dl, [CRT_COLS] ; test for column overflow
3603 <1> ; jne _set_cpos
3604 <1> ; 02/08/2022
3605 00002313 7402 <1> je short u13
3606 00002315 EB5C <1> jmp _set_cpos
3607 <1> u13:
3608 00002317 B200 <1> mov dl, 0 ; column = 0
3609 <1> u10: ; (line feed found)
3610 00002319 80FE18 <1> cmp dh, 25-1 ; check for last row
3611 0000231C 721F <1> jb short u6
3612 <1>
3613 <1> ; scroll required
3614 <1> u1:
3615 <1> ; SET CURSOR POSITION (04/12/2013)
3616 0000231E E850000000 <1> call _set_cpos
3617 <1>
3618 <1> ; determine value to fill with during scroll
3619 <1> u2:
3620 <1> ; bh = video page number
3621 <1>
3622 00002323 E8A3FEFFFF <1> call _read_ac_current ; 18/01/2016
3623 <1>
3624 <1> ; al = character, ah = attribute
3625 <1> ; bh = video page number
3626 <1> ; 18/11/2020
3627 00002328 88E3 <1> mov bl, ah ; color/attribute
3628 <1> u3:
3629 <1> ;;mov ax, 0601h ; scroll one line
3630 <1> ;;sub cx, cx ; upper left corner
3631 <1> ;;mov dh, 25-1 ; lower right row
3632 <1> ;;mov dl, [CRT_COLS]
3633 <1> ;mov dl, 80 ; lower right column
3634 <1> ;;dec dl
3635 <1> ;;mov dl, 79
3636 <1>
3637 <1> ;;call scroll_up ; 04/12/2013
3638 <1> ;; 11/03/2015
3639 <1> ; 02/09/2014
3640 <1> ;;mov cx, [crt_ulc] ; Upper left corner (0000h)
3641 <1> ;;mov dx, [crt_lrc] ; Lower right corner (184Fh)
3642 <1> ; 11/03/2015
3643 <1> ;sub cx, cx
3644 <1> ; 17/04/2021
3645 0000232A 29C9 <1> sub ecx, ecx

```

```

3646      <1>      ;mov     dx, 184Fh ; d1 = 79 (column), dh = 24 (row)
3647      <1>      ; 18/11/2020
3648      0000232C B618      <1>      mov     dh, 25-1
3649      0000232E 8A15[20680000] <1>      mov     d1, [CRT_COLS]
3650      00002334 FECA      <1>      dec     d1
3651      <1>      ;
3652      00002336 B001      <1>      mov     al, 1 ; scroll 1 line up
3653      <1>      ; ah = attribute
3654      <1>      ;mov     b1, al ; 12/05/2016
3655      00002338 E9F0FCFFFF <1>      jmp     _scroll_up ; 16/01/2016
3656      <1>      ;u4:
3657      <1>      ;;int     10h ; video-call return
3658      <1>      ; scroll up the screen
3659      <1>      ; tty return
3660      <1>      ;u5:
3661      <1>      ;retn ; return to the caller
3662      <1>
3663      <1>      u6: ; set-cursor-inc
3664      0000233D FEC6      <1>      inc     dh ; next row
3665      <1>      ; set cursor
3666      <1>      ;u7:
3667      <1>      ;;mov     ah, 02h
3668      <1>      ;;jmp     short u4 ; establish the new cursor
3669      <1>      ;call    _set_cpos
3670      <1>      ;jmp     short u5
3671      0000233F EB32      <1>      jmp     _set_cpos
3672      <1>
3673      <1>      ; check for control characters
3674      <1>      u8:
3675      <1>      ;je      short u9 ; 17/04/2021
3676      00002341 3C0A      <1>      cmp     al, 0Ah ; is it a line feed (0Ah)
3677      00002343 74D4      <1>      je      short u10
3678      00002345 3C07      <1>      cmp     al, 07h ; is it a bell
3679      00002347 747C      <1>      je      short u11
3680      00002349 3C08      <1>      cmp     al, 08h ; is it a backspace
3681      <1>      ;jne     short u0
3682      0000234B 741C      <1>      je      short bs ; 12/12/2013
3683      <1>      ; 12/12/2013 (tab stop)
3684      0000234D 3C09      <1>      cmp     al, 09h ; is it a tab stop
3685      0000234F 75B1      <1>      jne     short u0
3686      00002351 88D0      <1>      mov     al, d1
3687      <1>      ;cbw
3688      00002353 30E4      <1>      xor     ah, ah ; 09/12/2017
3689      00002355 B108      <1>      mov     cl, 8
3690      00002357 F6F1      <1>      div     cl
3691      00002359 28E1      <1>      sub     cl, ah
3692      <1>      ts:
3693      <1>      ; 02/09/2014
3694      <1>      ; 01/09/2014
3695      <1>      ;mov     al, 20h
3696      <1>      tsloop:
3697      <1>      ;push     cx
3698      <1>      ; 12/04/2021
3699      0000235B 51      <1>      push    ecx ; *
3700      <1>      ; 18/11/2020
3701      <1>      ;push     ax
3702      <1>      ;mov     bh, [ACTIVE_PAGE]
3703      <1>      ; 05/12/2020
3704      <1>      ;push     bx
3705      0000235C B020      <1>      mov     al, 20h ; al = space (blank)
3706      <1>      ; b1 = color/attribute
3707      0000235E E88BFFFFFF <1>      call    _write_tty_m3 ; 24/06/2016 (m3 -> _write_tty_m3)
3708      <1>      ; 05/12/2020
3709      <1>      ; bx is preserved in '_write_tty_m3'
3710      <1>      ; 18/11/2020
3711      <1>      ;pop      bx ; BL = color/attribute, Bh = video page
3712      <1>      ;pop      ax ; AL = character
3713      <1>      ; 12/04/2021
3714      <1>      ;pop      cx
3715      00002363 59      <1>      pop     ecx ; *
3716      00002364 FEC9      <1>      dec     cl
3717      00002366 75F3      <1>      jnz     short tsloop
3718      00002368 C3      <1>      retn
3719      <1>      bs:
3720      <1>      ; back space found
3721      <1>
3722      00002369 08D2      <1>      or      d1, d1 ; is it already at start of line
3723      <1>      ;je      short u7 ; set_cursor
3724      0000236B 7406      <1>      jz      short _set_cpos
3725      <1>      ;dec     dx ; no -- just move it back
3726      <1>      ; 17/04/2021
3727      0000236D FECA      <1>      dec     d1 ; move to 1 column back
3728      <1>      ;jmp     short u7
3729      0000236F EB02      <1>      jmp     short _set_cpos
3730      <1>
3731      <1>      ; carriage return found
3732      <1>      u9:
3733      00002371 B200      <1>      mov     d1, 0 ; move to first column
3734      <1>      ;jmp     short u7
3735      <1>      ;jmp     short _set_cpos ; 30/01/2016
3736      <1>
3737      <1>      ; line feed found
3738      <1>      ;u10:
3739      <1>      ; cmp     dh, 25-1 ; bottom of screen
3740      <1>      ; jne     short u6 ; no, just set the cursor
3741      <1>      ; jmp     u1 ; yes, scroll the screen
3742      <1>
3743      <1>      _set_cpos:
3744      <1>      ; 02/08/2022 - TRDOS 386 v2.0.5
3745      <1>      ; 17/04/2021 - TRDOS 386 v2.0.4
3746      <1>      ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
3747      <1>      ; 27/06/2015
3748      <1>      ; 01/09/2014
3749      <1>      ; 30/08/2014 (Retro UNIX 386 v1)
3750      <1>      ;
3751      <1>      ; 04/12/2013 - 12/12/2013 (Retro UNIX 8086 v1)
3752      <1>      ;
3753      <1>      ; VIDEO.ASM - 06/10/85 VIDEO DISPLAY BIOS
3754      <1>      ;
3755      <1>      ;-----
3756      <1>      ; SET_CPOS
3757      <1>      ; THIS ROUTINE SETS THE CURRENT CURSOR POSITION TO THE
3758      <1>      ; NEW X-Y VALUES PASSED
3759      <1>      ; INPUT
3760      <1>      ; DX - ROW,COLUMN OF NEW CURSOR
3761      <1>      ; BH - DISPLAY PAGE OF CURSOR
3762      <1>      ; OUTPUT
3763      <1>      ; CURSOR ID SET AT 6845 IF DISPLAY PAGE IS CURRENT DISPLAY
3764      <1>      ;-----
3765      <1>      ;
3766      00002373 BE[BE760100] <1>      mov     esi, CURSOR_POSN
3767      00002378 0FB6C7      <1>      movzx   eax, bh ; BH = video page number
3768      <1>      ; or      al, al
3769      <1>      ; jz      short _set_cpos_0

```



```

3770 0000237B D0E0      <1>      shl      al, 1      ; word offset
3771 0000237D 01C6      <1>      add      esi, eax
3772                                <1> ;_set_cpos_0:
3773 0000237F 668916      <1>      mov      [esi], dx ; save the pointer
3774 00002382 383D[CE760100] <1>      cmp      [ACTIVE_PAGE], bh
3775 00002388 7532      <1>      jne      short m17
3776                                <1> ;call m18 ; CURSOR SET
3777                                <1> ;m17: ; SET_CPOS_RETURN
3778                                <1> ; 01/09/2014
3779                                <1> ; retn
3780                                <1> ; DX = row/column
3781                                <1> m18:
3782 0000238A E852FCFFFF      <1>      call     position ; determine location in regen buffer
3783 0000238F 668B0D[BC760100] <1>      mov      cx, [CRT_START]
3784 00002396 6601C1      <1>      add      cx, ax ; add char position in regen buffer
3785                                <1> ; to the start address (offset) for this page
3786 00002399 66D1E9      <1>      shr      cx, 1 ; divide by 2 for char only count
3787 0000239C B40E      <1>      mov      ah, 14 ; register number for cursor
3788                                <1> ;call m16 ; output value to the 6845
3789                                <1> ;retn
3790                                <1>
3791                                <1> ;----- THIS ROUTINE OUTPUTS THE CX REGISTER
3792                                <1> ; TO THE 6845 REGISTERS NAMED IN (AH)
3793                                <1> m16:
3794 0000239E FA      <1>      cli
3795                                <1> ;mov dx, [addr_6845] ; address register
3796 0000239F 66BAD403      <1>      mov      dx, 03D4h ; I/O address of color card
3797 000023A3 88E0      <1>      mov      al, ah ; get value
3798 000023A5 EE      <1>      out      dx, al ; register set
3799                                <1> ;inc dx ; data register
3800                                <1> ; 17/04/2021
3801 000023A6 FEC2      <1>      inc      dl
3802 000023A8 EB00      <1>      jmp      $+2 ; i/o delay
3803 000023AA 88E8      <1>      mov      al, ch ; data
3804 000023AC EE      <1>      out      dx, al
3805                                <1> ;dec dx
3806                                <1> ; 17/04/2021
3807 000023AD FECA      <1>      dec      dl
3808 000023AF 88E0      <1>      mov      al, ah
3809 000023B1 FEC0      <1>      inc      al ; point to other data register
3810 000023B3 EE      <1>      out      dx, al ; set for second register
3811                                <1> ;inc dx
3812                                <1> ; 17/04/2021
3813 000023B4 FEC2      <1>      inc      dl
3814 000023B6 EB00      <1>      jmp      $+2 ; i/o delay
3815 000023B8 88C8      <1>      mov      al, cl ; second data value
3816 000023BA EE      <1>      out      dx, al
3817 000023BB FB      <1>      sti
3818                                <1> m17:
3819 000023BC C3      <1>      retn
3820                                <1>
3821                                <1> _beep:
3822                                <1> ; 12/02/2021 (TRDOS v2.0.3)
3823 000023BD FA      <1>      cli
3824 000023BE E811000000      <1>      call     beep
3825 000023C3 FB      <1>      sti
3826 000023C4 C3      <1>      retn
3827                                <1>
3828                                <1> beeper:
3829                                <1> ; 04/08/2016
3830                                <1> ; 12/05/2016 - TRDOS 386 (TRDOS v2.0)
3831                                <1> ; 30/08/2014 (Retro UNIX 386 v1)
3832                                <1> ; 18/01/2014
3833                                <1> ; 03/12/2013
3834                                <1> ; bell found
3835                                <1> u11:
3836 000023C5 FB      <1>      sti
3837 000023C6 3A3D[CE760100] <1>      cmp      bh, [ACTIVE_PAGE]
3838 000023CC 7552      <1>      jne      short u12 ; Do not sound the beep
3839                                <1> ; if it is not written on the active page
3840                                <1> beeper_gfx: ; 04/08/2016
3841 000023CE 66B93305      <1>      mov      cx, 1331 ; divisor for 896 hz tone
3842 000023D2 B31F      <1>      mov      bl, 31 ; set count for 31/64 second for beep
3843                                <1> ;call beep ; sound the pod bell
3844                                <1> ;jmp short u5 ; tty_return
3845                                <1> ;retn
3846                                <1>
3847                                <1> TIMER equ 040h ; 8254 TIMER - BASE ADDRESS
3848                                <1> PORT_B equ 061h ; PORT B READ/WRITE DIAGNOSTIC REGISTER
3849                                <1> GATE2 equ 00000001b ; TIMER 2 INPUT CATE CLOCK BIT
3850                                <1> SPK2equ 00000010b ; SPEAKER OUTPUT DATA ENABLE BIT
3851                                <1>
3852                                <1> beep:
3853                                <1> ; 12/02/2021
3854                                <1> ; 07/02/2015
3855                                <1> ; 30/08/2014 (Retro UNIX 386 v1)
3856                                <1> ; 18/01/2014
3857                                <1> ; 03/12/2013
3858                                <1> ;
3859                                <1> ; TEST4.ASM - 06/10/85 POST AND BIOS UTILITY ROUTINES
3860                                <1> ;
3861                                <1> ; ROUTINE TO SOUND THE BEEPER USING TIMER 2 FOR TONE
3862                                <1> ;
3863                                <1> ; ENTRY:
3864                                <1> ; (BL) = DURATION COUNTER ( 1 FOR 1/64 SECOND )
3865                                <1> ; (CX) = FREQUENCY DIVISOR (1193180/FREQUENCY) (1331 FOR 886 HZ)
3866                                <1> ; EXIT:
3867                                <1> ; (AX),(BL),(CX) MODIFIED.
3868                                <1>
3869 000023D4 9C      <1>      pushfd ; 18/01/2014 ; save interrupt status
3870 000023D5 FA      <1>      cli ; block interrupts during update
3871 000023D6 B0B6      <1>      mov      al, 10110110b ; select timer 2, lsb, msb binary
3872 000023D8 E643      <1>      out      TIMER+3, al ; write timer mode register
3873 000023DA EB00      <1>      jmp      $+2 ; I/O delay
3874 000023DC 88C8      <1>      mov      al, cl ; divisor for hz (low)
3875 000023DE E642      <1>      out      TIMER+2, AL ; write timer 2 count - lsb
3876 000023E0 EB00      <1>      jmp      $+2 ; I/O delay
3877 000023E2 88E8      <1>      mov      al, ch ; divisor for hz (high)
3878 000023E4 E642      <1>      out      TIMER+2, al ; write timer 2 count - msb
3879 000023E6 E461      <1>      in      al, PORT_B ; get current setting of port
3880 000023E8 88C4      <1>      mov      ah, al ; save that setting
3881 000023EA 0C03      <1>      or      al, GATE2+SPK2 ; gate timer 2 and turn speaker on
3882 000023EC E661      <1>      out      PORT_B, al ; and restore interrupt status
3883                                <1> ; 12/02/2021
3884 000023EE 9D      <1>      popfd ; 18/01/2014
3885                                <1> ;sti
3886                                <1> g7:
3887 000023EF B90B040000      <1>      mov      ecx, 1035 ; 1/64 second per count (bl)
3888 000023F4 E828000000      <1>      call     waitf ; delay count for 1/64 of a second
3889 000023F9 FECB      <1>      dec      bl ; go to beep delay 1/64 count
3890 000023FB 75F2      <1>      jnz      short g7 ; (bl) length count expired?
3891                                <1> ; no - continue beeping speaker
3892                                <1> ;
3892 000023FD 9C      <1>      pushfd ; 12/02/2021 ; save interrupt status
3893 000023FE FA      <1>      cli ; 18/01/2014 ; block interrupts during update

```

```

3894 000023FF E461      <1>      in      al, PORT_B      ; get current port value
3895                    <1>      ;or al, not (GATE2+SPK2) ; isolate current speaker bits in case
3896 00002401 0CFC      <1>      or       al, ~(GATE2+SPK2)
3897 00002403 20C4      <1>      and ah, al      ; someone turned them off during beep
3898 00002405 88E0      <1>      mov     al, ah      ; recover value of port
3899                    <1>      ;or al, not (GATE2+SPK2) ; force speaker data off
3900 00002407 0CFC      <1>      or       al, ~(GATE2+SPK2) ; isolate current speaker bits in case
3901 00002409 E661      <1>      out     PORT_B, al      ; and stop speaker timer
3902 0000240B 9D        <1>      popfd    ; 12/02/2021      ; restore interrupt flag state
3903                    <1>      ;sti
3904                    <1>      ;mov     ecx, 1035      ; force 1/64 second delay (short)
3905                    <1>      ; 17/04/2021
3906 0000240C 66B90B04    <1>      mov     cx, 1035
3907 00002410 E80C000000 <1>      call    waitf      ; minimum delay between all beeps
3908 00002415 9C        <1>      pushfd   ; save interrupt status
3909 00002416 FA        <1>      cli       ; block interrupts during update
3910 00002417 E461      <1>      in      al, PORT_B      ; get current port value in case
3911 00002419 2403      <1>      and     al, GATE2+SPK2 ; someone turned them on
3912 0000241B 08E0      <1>      or      al, ah      ; recover value of port_b
3913 0000241D E661      <1>      out     PORT_B, al      ; restore speaker status
3914 0000241F 9D        <1>      popfd    ; restore interrupt flag state
3915                    <1>      u12:
3916 00002420 C3        <1>      retn
3917                    <1>
3918                    <1>      REFRESH_BIT equ    00010000b      ; REFRESH TEST BIT
3919                    <1>
3920                    <1>      WAITF:
3921                    <1>      waitf:
3922                    <1>      ; 30/08/2014 (Retro UNIX 386 v1)
3923                    <1>      ; 03/12/2013
3924                    <1>      ;
3925                    <1>      ; push     ax      ; save work register (ah)
3926                    <1>      ;waitf1:
3927                    <1>      ; use timer 1 output bits
3928                    <1>      ; in      al, PORT_B      ; read current counter output status
3929                    <1>      ; and     al, REFRESH_BIT ; mask for refresh determine bit
3930                    <1>      ; cmp     al, ah      ; did it just change
3931                    <1>      ; je      short waitf1 ; wait for a change in output line
3932                    <1>      ;
3933                    <1>      ; mov     ah, al      ; save new lflag state
3934                    <1>      ; loop    waitf1      ; decrement half cycles till count end
3935                    <1>      ;
3936                    <1>      ; pop     ax      ; restore (ah)
3937                    <1>      ; retn     ; return (cx)=0
3938                    <1>
3939                    <1>      ; 06/02/2015 (unix386.s <-- dsectrm2.s)
3940                    <1>      ; 17/12/2014 (dsectrm2.s)
3941                    <1>      ; WAITF
3942                    <1>      ; /// IBM PC-XT Model 286 System BIOS Source Code - Test 4 - 06/10/85 ///
3943                    <1>
3944                    <1>      ---WAITF-----
3945                    <1>      ; FIXED TIME WAIT ROUTINE (HARDWARE CONTROLLED - NOT PROCESSOR)
3946                    <1>      ; ENTRY:
3947                    <1>      ; (CX) = COUNT OF 15.085737 MICROSECOND INTERVALS TO WAIT
3948                    <1>      ; MEMORY REFRESH TIMER 1 OUTPUT USED AS REFERENCE
3949                    <1>      ; EXIT:
3950                    <1>      ; AFTER (CX) TIME COUNT (PLUS OR MINUS 16 MICROSECONDS)
3951                    <1>      ; (CX) = 0
3952                    <1>      ;-----
3953                    <1>
3954                    <1>      ; Refresh period: 30 micro seconds (15-80 us)
3955                    <1>      ; (16/12/2014 - AWARD BIOS 1999 - ATORGs.ASM, WAIT_REFRESH)
3956                    <1>
3957                    <1>      ;WAITF:
3958                    <1>      ; push     ax      ; DELAY FOR (CX)*15.085737 US
3959                    <1>      ; 11/04/2021      ; SAVE WORK REGISTER (AH)
3960 00002421 50        <1>      push     eax
3961                    <1>      ; 16/12/2014
3962                    <1>      ; shr     cx, 1      ; convert to count of 30 micro seconds
3963 00002422 D1E9      <1>      shr     ecx, 1 ; 21/02/2015
3964                    <1>      ;17/12/2014
3965                    <1>      ;WAITF1:
3966                    <1>      ; in      al, PORT_B ;061h      ; READ CURRENT COUNTER OUTPUT STATUS
3967                    <1>      ; and     al, REFRESH_BIT ;00010000b ; MASK FOR REFRESH DETERMINE BIT
3968                    <1>      ; cmp     al, ah      ; DID IT JUST CHANGE
3969                    <1>      ; je      short WAITF1 ; WAIT FOR A CHANGE IN OUTPUT LINE
3970                    <1>      ; mov     ah, al      ; SAVE NEW FLAG STATE
3971                    <1>      ; loop    WAITF1      ; DECREMENT HALF CYCLES TILL COUNT END
3972                    <1>      ;
3973                    <1>      ; 17/12/2014
3974                    <1>
3975                    <1>      ; Modification from 'WAIT_REFRESH' procedure of AWARD BIOS - 1999
3976                    <1>
3977                    <1>      ;WAIT_REFRESH: Uses port 61, bit 4 to have CPU speed independent waiting.
3978                    <1>      ; INPUT: CX = number of refresh periods to wait
3979                    <1>      ; (refresh periods = 1 per 30 microseconds on most machines)
3980                    <1>
3981 00002424 E461      <1>      WR_STATE_0:
3982 00002426 A810      <1>      in      al, PORT_B      ; IN AL,SYS1
3983 00002428 74FA      <1>      test     al, 010h
3984                    <1>      JZ      short WR_STATE_0
3985 0000242A E461      <1>      WR_STATE_1:
3986 0000242C A810      <1>      in      al, PORT_B      ; IN AL,SYS1
3987 0000242E 75FA      <1>      test     al, 010h
3988 00002430 E2F2      <1>      jnz     short WR_STATE_1
3989                    <1>      loop    WR_STATE_0
3990                    <1>      ;
3991                    <1>      ; pop     ax      ; RESTORE (AH)
3992                    <1>      ; 11/04/2021
3993 00002432 58        <1>      pop     eax
3994                    <1>      retn
3995                    <1>      ; (CX) = 0
3996                    <1>
3997                    <1>      ; 03/08/2022
3998                    <1>      ; 02/08/2022 - TRDOS 386 Kernel v2.0.5
3999                    <1>      ; 09/07/2016
4000                    <1>      ; 01/07/2016
4001                    <1>      ; 24/06/2016
4002                    <1>      ; 23/06/2016 - TRDOS 386 (TRDOS v2.0)
4003                    <1>      ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
4004                    <1>
4005                    <1>      ;-----
4006                    <1>      ; WRITE_STRING
4007                    <1>      ; THIS ROUTINE WRITES A STRING OF CHARACTERS TO THE CRT.
4008                    <1>      ; INPUT
4009                    <1>      ; (AL) = WRITE STRING COMMAND 0 - 3
4010                    <1>      ; (BH) = DISPLAY PAGE (ACTIVE PAGE)
4011                    <1>      ; (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN
4012                    <1>      ; (DX) = CURSOR POSITION FOR START OF STRING WRITE
4013                    <1>      ; (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0 OR (AL) = 1
4014                    <1>      ; (EBP) = SOURCE STRING OFFSET
4015                    <1>      ; OUTPUT
4016                    <1>      ; NONE
4017                    <1>      ;-----
4018                    <1>
4019                    <1>      ; AL = 00h: Assign all characters the attribute in BL; do not update cursor
4020                    <1>      ; AL = 01h: Assign all characters the attribute in BL; update cursor

```

```

4018 <1> ; AL = 02h: Use attributes in string; do not update cursor
4019 <1> ; AL = 03h: Use attributes in string; update cursor
4020 <1>
4021 <1> WRITE_STRING:
4022 <1> ; 03/08/2022
4023 <1> ; 02/08/2022
4024 <1> ; 12/09/2016
4025 <1> ; 09/07/2016
4026 <1> ;cmp byte [CRT_MODE], 7 ; 6?!
4027 <1> ;ja VIDEO_RETURN ; not a valid function for VGA modes
4028 <1>
4029 00002434 A2[34830100] <1> mov [w_str_cmd], al ; save (AL) command
4030 00002439 3C04 <1> cmp al, 4 ; TEST FOR INVALID WRITE STRING OPTION
4031 <1> ;jnb VIDEO_RETURN ; IF OPTION INVALID THEN RETURN
4032 <1> ; 02/08/2022
4033 0000243B 7361 <1> jnb short P55
4034 <1>
4035 <1> ;jcxz VIDEO_RETURN ; IF ZERO LENGTH STRING THEN RETURN
4036 <1>
4037 0000243D 67E35E <1> jcxz P55 ; 01/07/2016
4038 <1>
4039 <1> ; 01/07/2016
4040 <1> ;and ecx, 0FFFFh
4041 <1> ; ecx = byte count
4042 <1> ;push ecx
4043 00002440 89EE <1> mov esi, ebp ; user buffer
4044 00002442 BF00000700 <1> mov edi, Cluster_Buffer ; system buffer
4045 00002447 E809E70000 <1> call transfer_from_user_buffer
4046 <1> ;pop ecx
4047 <1> ;jc VIDEO_RETURN
4048 <1> ; 02/08/2022
4049 0000244C 7250 <1> jc short P55
4050 <1> ; ecx = transfer (byte) count = character count
4051 0000244E BD00000700 <1> mov ebp, Cluster_Buffer
4052 <1> ; 12/09/2016
4053 00002453 803D[1E680000]07 <1> cmp byte [CRT_MODE], 7 ; 6?!
4054 <1> ;ja vga_write_string
4055 <1> ; 02/08/2022
4056 0000245A 7605 <1> jna short P57
4057 0000245C E99C000000 <1> jmp vga_write_string
4058 <1> P57:
4059 00002461 0FB6F7 <1> movzx esi, bh ; GET CURRENT CURSOR PAGE
4060 <1> ;sal si, 1 ; CONVERT TO PAGE OFFSET (SI = PAGE)
4061 <1> ; 02/08/2022
4062 00002464 D1E6 <1> sal esi, 1
4063 <1> ; *****
4064 00002466 66FFB6[BE760100] <1> push word [esi+CURSOR_POSN]; SAVE CURRENT CURSOR POSITION IN STACK
4065 <1>
4066 <1> ;mov ax, 0200h ; SET NEW CURSOR POSITION
4067 <1> ;int 10h
4068 <1> P50next:
4069 0000246D 51 <1> push ecx ; *****
4070 0000246E 53 <1> push ebx ; *** ; 18/11/2020
4071 0000246F 56 <1> push esi ; **
4072 00002470 52 <1> push edx ; *
4073 00002471 E8FDFEFFFF <1> call _set_cpos
4074 <1> P50:
4075 00002476 8A4500 <1> mov al, [ebp] ; GET CHARACTER FROM INPUT STRING
4076 00002479 45 <1> inc ebp ; BUMP POINTER TO CHARACTER
4077 <1>
4078 <1> ;----- TEST FOR SPECIAL CHARACTER'S
4079 <1>
4080 0000247A 3C08 <1> cmp al, 08h ; IS IT A BACKSPACE
4081 0000247C 7410 <1> je short P51 ; BACK_SPACE
4082 0000247E 3C0D <1> cmp al, 0Dh ; CR
4083 00002480 740C <1> je short P51 ; IS IT CARRIAGE RETURN
4084 00002482 3C0A <1> cmp al, 0Ah ; LF
4085 00002484 7408 <1> je short P51 ; CAR_RET
4086 <1> ; 18/11/2020
4087 00002486 3C09 <1> cmp al, 09h ; is it a tab stop
4088 00002488 7404 <1> je short P51
4089 <1>
4090 0000248A 3C07 <1> cmp al, 07h ; IS IT A BELL
4091 0000248C 7515 <1> jne short P52 ; IF NOT THEN DO WRITE CHARACTER
4092 <1> P51:
4093 <1> ;mov ah, 0Eh ; TTY_CHARACTER_WRITE
4094 <1> ;int 10h ; WRITE TTY CHARACTER TO THE CRT
4095 <1>
4096 0000248E E85BFEFFFF <1> call _write_tty_m3
4097 <1>
4098 00002493 5A <1> pop edx ; *
4099 00002494 5E <1> pop esi ; **
4100 <1>
4101 00002495 668B96[BE760100] <1> mov dx, [esi+CURSOR_POSN] ; GET CURRENT CURSOR POSITION
4102 0000249C EB44 <1> jmp short P54 ; SET CURSOR POSITION AND CONTINUE
4103 <1> P55:
4104 0000249E E9D5F6FFFF <1> jmp VIDEO_RETURN
4105 <1> P52:
4106 <1> ;mov cx, 1 ; SET CHARACTER WRITE AMOUNT TO ONE
4107 <1> ; 02/08/2022
4108 000024A3 29C9 <1> sub ecx, ecx
4109 000024A5 FEC1 <1> inc cl
4110 <1> ; ecx = 1
4111 000024A7 803D[34830100]02 <1> cmp byte [w_str_cmd], 2 ; IS THE ATTRIBUTE IN THE STRING
4112 000024AE 7204 <1> jb short P53 ; IF NOT THEN SKIP
4113 000024B0 8A5D00 <1> mov bl, [ebp] ; ELSE GET NEW ATTRIBUTE
4114 000024B3 45 <1> inc ebp ; BUMP STRING POINTER
4115 <1> P53:
4116 <1> ;mov ah, 09h ; GOT_CHARACTER
4117 <1> ;int 10h ; WRITE CHARACTER TO THE CRT
4118 <1>
4119 000024B4 E8A1FDFFFF <1> call _write_c_current
4120 <1>
4121 000024B9 5A <1> pop edx ; *
4122 <1>
4123 <1> ; 05/12/2020
4124 <1> ; bx is preserved in '_write_c_current'
4125 <1> ; 18/11/2020
4126 <1> ;mov ebx, [esp+4] ; ***
4127 <1>
4128 000024BA 0FB6F7 <1> movzx esi, bh ; video page number (0 to 7)
4129 000024BD 889E[27680000] <1> mov [esi+chr_attrib], bl ; color/attribute
4130 <1>
4131 000024C3 FEC2 <1> inc dl ; INCREMENT COLUMN COUNTER
4132 000024C5 3A15[20680000] <1> cmp dl, [CRT_COLS] ; IF COLS ARE WITHIN RANGE FOR THIS MODE
4133 <1> ;jb short P54 ; THEN GO TO COLUMNS SET
4134 000024CB 7214 <1> jb short P56 ; 05/12/2020
4135 000024CD FEC6 <1> inc dh ; BUMP ROW COUNTER BY ONE
4136 000024CF 28D2 <1> sub dl, dl ; SET COLUMN COUNTER TO ZERO
4137 000024D1 80FE19 <1> cmp dh, 25 ; IF ROWS ARE LESS THAN 25 THEN
4138 <1> ;jb short P54 ; GO TO ROWS_COLUMNS_SET
4139 000024D4 720B <1> jb short P56 ; 05/12/2020
4140 <1>
4141 <1> ; 18/11/2020

```

```

4142      <1>      ;mov     ax, 0E0Ah          ; ELSE SCROLL SCREEN
4143      <1>      ;int      10h            ; RESET ROW COUNTER TO 24
4144      <1>
4145      <1>      ; 18/11/2020
4146 000024D6 B00A      <1>      mov     al, 0Ah ; line feed
4147      <1>
4148 000024D8 E811FEFFFF <1>      call    _write_tty_m3
4149      <1>
4150 000024DD 66BA0018   <1>      mov     dx, 1800h          ; Column = 0, Row = 24
4151      <1> P56:
4152      <1>      ; 05/12/2020
4153      <1>      ; 18/11/2020
4154 000024E1 5E        <1>      pop     esi ; **
4155      <1> P54:
4156      <1>      ;mov     ax, 0200h          ; ROW_COLUMNS_SET
4157      <1>      ;int      10h            ; SET NEW CURSOR POSITION COMMAND
4158      <1>      ; ESTABLISH NEW CURSOR POSITION
4159      <1>      ; 18/11/2020
4160 000024E2 5B        <1>      pop     ebx ; ***
4161 000024E3 59        <1>      pop     ecx ; ****
4162      <1>
4163      <1>      ;loop    P50                ; DO IT ONCE MORE UNTIL (CX) = ZERO
4164 000024E4 6649      <1>      dec     cx
4165 000024E6 7585      <1>      jnz     short P50next
4166      <1>
4167 000024E8 665A      <1>      pop     dx ; *****
4168      <1>      ; RESTORE OLD CURSOR COORDINATES
4169 000024EA F605[34830100]01 <1>      test    byte [w_str_cmd], 1 ; IF CURSOR WAS NOT TO BE MOVED
4170      <1>      ;jnz     VIDEO_RETURN      ; THEN EXIT WITHOUT RESETTNG OLD VALUE
4171      <1>      ; 03/08/2022
4172 000024F1 7505      <1>      jnz     short P58
4173      <1>
4174      <1>      ;mov     ax, 0200h          ; ELSE RESTORE OLD CURSOR POSITION
4175      <1>      ;int      10h
4176      <1>      ; DONE - EXIT WRITE STRING
4177 000024F3 E87BFEFFFF <1>      call    _set_cpos
4178      <1> P58:
4179 000024F8 E97BF6FFFF <1>      jmp     VIDEO_RETURN      ; RETURN TO CALLER
4180      <1>
4181      <1> vga_write_string:
4182      <1>      ; 04/08/2022 - TRDOS 386 kernel v2.0.5
4183      <1>      ; 12/09/2016 - TRDOS 386 (TRDOS v2.0)
4184      <1>      ;
4185      <1>      ; derived from 'Plex86/Bochs VGABios' source code
4186      <1>      ; vgabios-0.7a (2011)
4187      <1>      ; by the LGPL VGABios developers Team (2001-2008)
4188      <1>      ; 'vgabios.c', 'biosfn_write_string'
4189      <1>
4190      <1>      ; INPUT
4191      <1>      ; (AL) = WRITE STRING COMMAND 0 - 3
4192      <1>      ; (BH) = DISPLAY PAGE (ACTIVE PAGE)
4193      <1>      ; (CX) = COUNT OF CHARACTERS TO WRITE, IF (CX) = 0 THEN RETURN
4194      <1>      ; (DX) = CURSOR POSITION FOR START OF STRING WRITE
4195      <1>      ; (BL) = ATTRIBUTE OF CHARACTER TO WRITE IF (AL) = 0 OR (AL) = 1
4196      <1>      ; (EBP) = SOURCE STRING OFFSET
4197      <1>      ; OUTPUT
4198      <1>      ; NONE
4199      <1>      ;-----;
4200      <1>
4201      <1>      ; AL = 00h: Assign all characters the attribute in BL; do not update cursor
4202      <1>      ; AL = 01h: Assign all characters the attribute in BL; update cursor
4203      <1>      ; AL = 02h: Use attributes in string; do not update cursor
4204      <1>      ; AL = 03h: Use attributes in string; update cursor
4205      <1>
4206      <1>      ; biosfn_write_string(GET_AL(),GET_BH(),GET_BL(),CX,GET_DH(),GET_DL(),ES,BP);
4207      <1>      ; static void biosfn_write_string (flag,page,attr,count,row,col,seg,offset)
4208      <1>
4209      <1>      ; // Read curs info for the page
4210      <1>      ; biosfn_get_cursor_pos(page,&dummy,&oldcurs);
4211      <1>      ; bh = video page = 0
4212      <1>      ;movzx esi, word [CURSOR_POSN] ; current cursor position for video page 0
4213      <1>
4214      <1>      ; // if row=0xff special case : use current cursor position
4215      <1>      ; if(row==0xff)
4216      <1>      ; {col=oldcurs&0x00ff;
4217      <1>      ; row=(oldcurs&0xff00)>>8;
4218      <1>      ; }
4219      <1>
4220      <1>      ;mov     al, [w_str_cmd]
4221      <1>
4222 000024FD 80FEFF      <1>      cmp     dh, 0FFh
4223 00002500 7407      <1>      je      short vga_wstr_1 ; user current cursor position
4224      <1> vga_wstr_0:
4225      <1>      ; set cursor position
4226 00002502 668915[BE760100] <1>      mov     [CURSOR_POSN], dx ; save cursor pos for pg 0
4227      <1> vga_wstr_1:
4228 00002509 66FF35[BE760100] <1>      push    word [CURSOR_POSN] ; *
4229      <1>
4230      <1>      ; ebp = string offset in system buffer (user buffer was copied to)
4231      <1>
4232      <1>      ; while(count--!=0)
4233      <1>      ; {
4234      <1>      ;     car=read_byte(seg,offset++);
4235      <1>      ;     if((flag&0x02)!=0)
4236      <1>      ;     attr=read_byte(seg,offset++);
4237      <1>      ;     biosfn_write_teletype(car,page,attr,WITH_ATTR);
4238      <1>      ; }
4239      <1>
4240      <1>      ;push     eax ; **
4241      <1>      ;test     al, 2
4242 00002510 F605[34830100]02 <1>      test    byte [w_str_cmd], 2
4243 00002517 751D      <1>      jnz     short vga_wstr_3
4244 00002519 881D[CF760100] <1>      mov     [ccolor], bl
4245      <1> vga_wstr_2:
4246      <1>      push    ecx
4247 00002520 8A4500      <1>      mov     al, [ebp]
4248 00002523 E88F0A0000      <1>      call    vga_write_teletype
4249 00002528 59        <1>      pop     ecx
4250 00002529 6649      <1>      dec     cx
4251 0000252B 741E      <1>      jz      short vga_wstr_4
4252 0000252D 45        <1>      inc     ebp
4253 0000252E 8A1D[CF760100] <1>      mov     bl, [ccolor]
4254 00002534 EBE9      <1>      jmp     short vga_wstr_2
4255      <1> vga_wstr_3:
4256      <1>      push    ecx
4257 00002537 8A4500      <1>      mov     al, [ebp]
4258 0000253A 45        <1>      inc     ebp
4259 0000253B 8A5D00      <1>      mov     bl, [ebp]
4260 0000253E E8740A0000      <1>      call    vga_write_teletype
4261 00002543 59        <1>      pop     ecx
4262 00002544 6649      <1>      dec     cx
4263 00002546 7403      <1>      jz      short vga_wstr_4
4264 00002548 45        <1>      inc     ebp
4265 00002549 EBEB      <1>      jmp     short vga_wstr_3

```

```

4266 <1> vga_wstr_4:
4267 <1> ; // Set back curs pos
4268 <1> ; if((flag&0x01)==0)
4269 <1> ; biosfn_set_cursor_pos(page,oldcurs);
4270 <1> ; }
4271 <1> ;pop     eax ; **
4272 0000254B 665A <1> pop     dx ; word [CURSOR_POSN] ; *
4273 <1> ;test    al, 1
4274 0000254D F605[34830100]01 <1> test    byte [w_str_cmd], 1
4275 <1> ;jnz     VIDEO_RETURN
4276 <1> ; 04/08/2022
4277 00002554 7507 <1> jnz     short vga_wstr_5
4278 00002556 668915[BE760100] <1> mov     [CURSOR_POSN], dx
4279 <1> vga_wstr_5:
4280 0000255D E916F6FFFF <1> jmp     VIDEO_RETURN
4281 <1>
4282 <1> ; 03/08/2022 - TRDOS 386 Kernel v2.0.5
4283 <1> ; 07/07/2016
4284 <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
4285 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
4286 <1> ;-----
4287 <1> ; SCROLL UP
4288 <1> ; THIS ROUTINE SCROLLS UP THE INFORMATION ON THE CRT
4289 <1> ; ENTRY ---
4290 <1> ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
4291 <1> ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
4292 <1> ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
4293 <1> ; BH = FILL VALUE FOR BLANKED LINES
4294 <1> ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
4295 <1> ; DS = DATA SEGMENT
4296 <1> ; ES = REGEN SEGMENT
4297 <1> ; EXIT --
4298 <1> ; NOTHING, THE SCREEN IS SCROLLED
4299 <1> ;-----
4300 <1>
4301 <1> ; cl = upper left column
4302 <1> ; ch = upper left row
4303 <1> ; dl = lower righth column
4304 <1> ; dh = lower right row
4305 <1> ;
4306 <1> ; al = line count (AL=0 means blank entire fields)
4307 <1> ; bl = fill value for blanked lines
4308 <1> ; bh = unused
4309 <1>
4310 <1> GRAPHICS_UP:
4311 <1> ; 07/07/2016
4312 <1> ; AH = Current video mode, [CRT_MODE]
4313 00002562 80FC07 <1> cmp     ah, 7
4314 00002565 7762 <1> ja      short vga_graphics_up
4315 <1> ;je      n0
4316 <1>
4317 00002567 88C7 <1> mov     bh, al ; save line count in BH
4318 <1> ;mov     ax, cx ; GET UPPER LEFT POSITION INTO AX REG
4319 <1> ; 03/08/2022
4320 00002569 89C8 <1> mov     eax, ecx
4321 <1>
4322 <1> ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
4323 <1> ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
4324 <1>
4325 0000256B E8BA050000 <1> call    GRAPH_POSN
4326 <1> ;movzx   edi, ax ; SAVE RESULT AS DESTINATION ADDRESS
4327 <1> ; 03/08/2022
4328 00002570 89C7 <1> mov     edi, eax
4329 <1>
4330 <1> ;----- DETERMINE SIZE OF WINDOW
4331 <1>
4332 00002572 6629CA <1> sub     dx, cx
4333 00002575 6681C20101 <1> add     dx, 101h ; ADJUST VALUES
4334 0000257A C0E602 <1> sal     dh, 2 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
4335 <1> ; AND EVEN/ODD ROWS
4336 <1> ;----- DETERMINE CRT MODE
4337 <1>
4338 0000257D 803D[1E680000]06 <1> cmp     byte [CRT_MODE], 6 ; TEST FOR MEDIUM RES
4339 00002584 7304 <1> jnc     short _R7_ ; FIND_SOURCE
4340 <1>
4341 <1> ;----- MEDIUM RES UP
4342 00002586 D0E2 <1> sal     dl, 1 ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
4343 <1> ;sal     di, 1 ; OFFSET *2 SINCE 2 BYTES/CHAR
4344 <1> ; 03/08/2022
4345 00002588 D1E7 <1> sal     edi, 1
4346 <1>
4347 <1> ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
4348 <1> _R7_: ; FIND_SOURCE
4349 0000258A 81C700800B00 <1> add     edi, 0B8000h
4350 00002590 C0E702 <1> sal     bh, 2 ; multiply number of lines by 4
4351 00002593 7430 <1> jz      short _R11 ; IF ZERO, THEN BLANK ENTIRE FIELD
4352 00002595 B050 <1> mov     al, 80 ; 80 BYTES/ROW
4353 00002597 F6E7 <1> mul     bh ; determine offset to source
4354 <1> ;movzx   esi, ax ; offset to source
4355 <1> ; 03/08/2022
4356 00002599 89C6 <1> mov     esi, eax
4357 0000259B 01FE <1> add     esi, edi ; SET UP SOURCE
4358 0000259D 88F4 <1> mov     ah, dh ; NUMBER OF ROWS IN FIELD
4359 0000259F 28FC <1> sub     ah, bh ; determine number to move
4360 <1>
4361 <1> ;----- LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
4362 <1> _R8: ; ROW_LOOP
4363 000025A1 E8FD030000 <1> call    _R17 ; MOVE ONE ROW
4364 000025A6 6681EEB01F <1> sub     si, 2000h-80 ; MOVE TO NEXT ROW
4365 000025AB 6681EFB01F <1> sub     di, 2000h-80
4366 000025B0 FECC <1> dec     ah ; NUMBER OF ROWS TO MOVE
4367 000025B2 75ED <1> jnz     short _R8 ; CONTINUE TILL ALL MOVED
4368 <1>
4369 <1> ;----- FILL IN THE VACATED LINE(S)
4370 <1> _R9: ; CLEAR ENTRY
4371 000025B4 88D8 <1> mov     al, bl ; attribute to fill with
4372 <1> _R10_:
4373 000025B6 E804040000 <1> call    _R18 ; CLEAR THAT ROW
4374 000025BB 6681EFB01F <1> sub     di, 2000h-80 ; POINT TO NEXT LINE
4375 000025C0 F6CF <1> dec     bh ; number of lines to fill
4376 000025C2 75F2 <1> jnz     short _R10_ ; CLEAR LOOP
4377 000025C4 C3 <1> retn ; EVERYTHING DONE
4378 <1>
4379 <1> _R11: ; BLANK_FIELD
4380 000025C5 88F7 <1> mov     bh, dh ; set blank count to everything in field
4381 000025C7 EBEB <1> jmp     short _R9 ; CLEAR THE FIELD
4382 <1>
4383 <1> vga_graphics_up:
4384 <1> ; 03/08/2022 - TRDOS 386 kernel v2.0.5
4385 <1> ; 12/04/2021
4386 <1> ; 08/08/2016
4387 <1> ; 07/08/2016
4388 <1> ; 04/08/2016
4389 <1> ; 01/08/2016

```

```

4390      <1>      ; 31/07/2016
4391      <1>      ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
4392      <1>      ;
4393      <1>      ; derived from 'Plex86/Bochs VGABios' source code
4394      <1>      ; vgabios-0.7a (2011)
4395      <1>      ; by the LGPL VGABios developers Team (2001-2008)
4396      <1>      ; 'vgabios.c', 'biosfn_scroll'
4397      <1>      ;
4398      <1>      ;
4399      <1>      ; cl = upper left column
4400      <1>      ; ch = upper left row
4401      <1>      ; dl = lower right column
4402      <1>      ; dh = lower right row
4403      <1>      ;
4404      <1>      ; al = line count (AL=0 means blank entire fields)
4405      <1>      ; bl = fill value for blanked lines
4406      <1>      ; bh = unused
4407      <1>      ;
4408      <1>      ; ah = [CRT_MODE], current video mode
4409      <1>      ;
4410      000025C9 88C7      <1>      mov     bh, al ; 31/07/2016
4411      000025CB BE[42680000] <1>      mov     esi, vga_g_modes
4412      000025D0 89F7      <1>      mov     edi, esi
4413      000025D2 83C708    <1>      add     edi, vga_g_mode_count
4414      <1>      vga_g_up_0:
4415      000025D5 AC        <1>      lodsb
4416      000025D6 38E0      <1>      cmp     al, ah ; [CRT_MODE]
4417      000025D8 7405      <1>      je      short vga_g_up_1
4418      000025DA 39FE      <1>      cmp     esi, edi
4419      000025DC 72F7      <1>      jb      short vga_g_up_0
4420      <1>      ;xor     bh, bh ; 31/07/2016)
4421      000025DE C3        <1>      retn     ; nothing to do
4422      <1>      vga_g_up_1:
4423      000025DF 88F8      <1>      mov     al, bh ; 31/07/2016
4424      000025E1 83C64F    <1>      add     esi, vga_g_memmodel - (vga_g_modes + 1)
4425      <1>      ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
4426      <1>      ;
4427      <1>      ; if(rlr>=nbrows)rlr=nbrows-1;
4428      <1>      ; if(clr>=nbcols)clr=nbcols-1;
4429      <1>      ; if(nblines>nbrows)nblines=0;
4430      <1>      ; cols=clr-cul+1;
4431      <1>      ;
4432      000025E4 3A35[26680000] <1>      cmp     dh, [VGA_ROWS]
4433      000025EA 7208      <1>      jb      short vga_g_up_2
4434      000025EC 8A35[26680000] <1>      mov     dh, [VGA_ROWS]
4435      000025F2 FECE      <1>      dec     dh
4436      <1>      vga_g_up_2:
4437      000025F4 3A15[20680000] <1>      cmp     dl, [CRT_COLS] ; = [VGA_COLS]
4438      000025FA 7208      <1>      jb      short vga_g_up_3
4439      000025FC 8A15[20680000] <1>      mov     dl, [CRT_COLS]
4440      00002602 FECA      <1>      dec     dl
4441      <1>      vga_g_up_3:
4442      00002604 3A05[26680000] <1>      cmp     al, [VGA_ROWS]
4443      0000260A 7602      <1>      jna     short vga_g_up_4
4444      0000260C 28C0      <1>      sub     al, al ; 0
4445      <1>      vga_g_up_4:
4446      0000260E 88D7      <1>      mov     bh, dl ; clr
4447      00002610 28CF      <1>      sub     bh, cl ; cul
4448      00002612 FEC7      <1>      inc     bh ; cols = clr-cul+1
4449      <1>      ;
4450      00002614 20C0      <1>      and     al, al ; nblines = 0
4451      00002616 7559      <1>      jnz     short vga_g_up_6
4452      00002618 20ED      <1>      and     ch, ch ; rul = 0
4453      0000261A 7555      <1>      jnz     short vga_g_up_6
4454      0000261C 20C9      <1>      and     cl, cl ; cul = 0
4455      0000261E 7551      <1>      jnz     short vga_g_up_6
4456      <1>      ;
4457      <1>      ;push    ax
4458      <1>      ; 12/04/2021
4459      00002620 50        <1>      push    eax
4460      00002621 A0[26680000] <1>      mov     al, [VGA_ROWS]
4461      00002626 FEC8      <1>      dec     al
4462      00002628 38C6      <1>      cmp     dh, al ; rlr = nbrows-1
4463      0000262A 7545      <1>      jne     short vga_g_up_5
4464      0000262C A0[20680000] <1>      mov     al, [CRT_COLS] ; = VGA_COLS
4465      00002631 FEC8      <1>      dec     al
4466      00002633 38C2      <1>      cmp     dl, al ; clr = nbcols-1
4467      <1>      ;jne     short vga_g_up_5
4468      <1>      ;pop     ax
4469      <1>      ; 12/04/2021
4470      00002635 58        <1>      pop     eax
4471      00002636 7539      <1>      jne     short vga_g_up_5
4472      <1>      ;
4473      00002638 66B80502 <1>      mov     ax, 0205h
4474      0000263C 66BACE03 <1>      mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
4475      00002640 66EF      <1>      out     dx, ax
4476      00002642 A0[26680000] <1>      mov     al, [VGA_ROWS]
4477      00002647 8A25[20680000] <1>      mov     ah, [CRT_COLS] ; = [VGA_COLS]
4478      0000264D F6E4      <1>      mul     ah
4479      0000264F 0FB7D0    <1>      movzx   edx, ax
4480      <1>      ; 08/08/2016
4481      00002652 0FB605[22680000] <1>      movzx   eax, byte [CHAR_HEIGHT]
4482      00002659 F7E2      <1>      mul     edx
4483      <1>      ; eax = byte count
4484      0000265B 89C1      <1>      mov     ecx, eax
4485      <1>      ; 07/08/2016
4486      <1>      ;shl     dx, 3 ; * 8 ; * [CHAR_HEIGHT]
4487      <1>      ;mov     ecx, edx
4488      0000265D 88D8      <1>      mov     al, bl ; fill value for blanked lines
4489      0000265F BF0000A00 <1>      mov     edi, 0A0000h
4490      00002664 F3AA      <1>      rep     stosb
4491      <1>      ;
4492      <1>      ;mov     ax, 5
4493      <1>      ; 03/08/2022
4494      00002666 30E4      <1>      xor     ah, ah
4495      00002668 B005      <1>      mov     al, 5
4496      <1>      ;
4497      0000266A 66BACE03 <1>      mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
4498      0000266E 66EF      <1>      out     dx, ax ; 0005h
4499      <1>      ;
4500      00002670 C3        <1>      retn
4501      <1>      ;
4502      <1>      vga_g_up_5:
4503      <1>      ;pop     ax
4504      <1>      ; 12/04/2021
4505      <1>      ;pop     eax
4506      <1>      ;
4507      <1>      vga_g_up_6:
4508      <1>      ; [ESI] = VGA memory model number for current video mode
4509      <1>      ;
4510      <1>      ; LINEAR8 equ 5
4511      <1>      ; PLANAR4 equ 4
4512      <1>      ; PLANAR1 equ 3
4513      <1>      ;

```

```

4514 00002671 803E04    <1>    cmp     byte [esi], PLANAR4
4515 00002674 7424    <1>    je      short vga_g_up_planar
4516 00002676 803E03    <1>    cmp     byte [esi], PLANAR1
4517 00002679 741F    <1>    je      short vga_g_up_planar
4518    <1> vga_g_up_linear8:
4519    <1>    ; 07/07/2016 (TEMPORARY)
4520    <1>    ;
4521    <1>    ; cl = upper left column ; cul
4522    <1>    ; ch = upper left row ; rul
4523    <1>    ; dl = lower right column ; clr
4524    <1>    ; dh = lower right row ; rlr
4525    <1>
4526    <1> vga_g_up_l0:
4527    <1>    ;{for(i=rul;i<=rlr;i++)
4528    <1>    ; if((i+nblines>rlr)|| (nblines==0))
4529 0000267B 08C0    <1>    or      al, al
4530 0000267D 7414    <1>    jz      short vga_g_up_l2
4531 0000267F 88C4    <1>    mov     ah, al
4532 00002681 00EC    <1>    add     ah, ch ; i+nblines
4533    <1>    ;jc     short vga_g_up_l2
4534 00002683 38F4    <1>    cmp     ah, dh
4535 00002685 770C    <1>    ja      short vga_g_up_l2
4536    <1>    ; else
4537    <1>    ; vgamem_copy_p14(cul,i+nblines,i,cols,nbcols,height);
4538 00002687 E8F2000000    <1>    call    vgamem_copy_l8
4539    <1> vga_g_up_l1:
4540    <1>    inc     ch
4541 0000268E 38F5    <1>    cmp     ch, dh
4542 00002690 76E9    <1>    jna     short vga_g_up_l0
4543 00002692 C3    <1>    retn
4544    <1> vga_g_up_l2:
4545    <1>    ; vgamem_fill_p14(cul,i,cols,nbcols,height,attr);
4546 00002693 E84C010000    <1>    call    vgamem_fill_l8
4547 00002698 EBF2    <1>    jmp     short vga_g_up_l1
4548    <1>
4549    <1> vga_g_up_planar:
4550    <1>    ; cl = upper left column ; cul
4551    <1>    ; ch = upper left row ; rul
4552    <1>    ; dl = lower right column ; clr
4553    <1>    ; dh = lower right row ; rlr
4554    <1> vga_g_up_p10:
4555    <1>    ;{for(i=rul;i<=rlr;i++)
4556    <1>    ; if((i+nblines>rlr)|| (nblines==0))
4557 0000269A 20C0    <1>    and     al, al
4558 0000269C 7414    <1>    jz      short vga_g_up_p12
4559 0000269E 88C4    <1>    mov     ah, al
4560 000026A0 00EC    <1>    add     ah, ch ; i+nblines
4561    <1>    ;jc     short vga_g_up_p12
4562 000026A2 38F4    <1>    cmp     ah, dh
4563 000026A4 770C    <1>    ja      short vga_g_up_p12
4564    <1>    ; else
4565    <1>    ; vgamem_copy_p14(cul,i+nblines,i,cols,nbcols,height);
4566 000026A6 E80E000000    <1>    call    vgamem_copy_p14
4567    <1> vga_g_up_p11:
4568 000026AB FEC5    <1>    inc     ch
4569 000026AD 38F5    <1>    cmp     ch, dh
4570 000026AF 76E9    <1>    jna     short vga_g_up_p10
4571 000026B1 C3    <1>    retn
4572    <1> vga_g_up_p12:
4573    <1>    ; vgamem_fill_p14(cul,i,cols,nbcols,height,attr);
4574 000026B2 E870000000    <1>    call    vgamem_fill_p14
4575 000026B7 EBF2    <1>    jmp     short vga_g_up_p11
4576    <1>
4577    <1> vgamem_copy_p14:
4578    <1>    ; 08/08/2016
4579    <1>    ; 07/08/2016
4580    <1>    ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
4581    <1>    ;
4582    <1>    ; derived from 'Plex86/Bochs VGABios' source code
4583    <1>    ; vgabios-0.7a (2011)
4584    <1>    ; by the LGPL VGABios developers Team (2001-2008)
4585    <1>    ; 'vgabios.c', 'vgamem_copy_p14'
4586    <1>    ;
4587    <1>    ; vgamem_copy_p14(xstart,ysrc,ydest,cols,nbcols,height)
4588    <1>    ; cl = xstart, ah = ysrc (i+nblines), ch = ydest (i),
4589    <1>    ; bh = cols, [CRT_COLS] = nbcols, [CHAR_HEIGHT] = height
4590    <1>
4591    <1>    ; src=ysrc*height*nbcols+xstart;
4592    <1>    ; dest=ydest*height*nbcols+xstart;
4593    <1>
4594 000026B9 52    <1>    push    edx
4595 000026BA 50    <1>    push    eax
4596    <1>
4597    <1>    ; outw(VGAREG_GRDC_ADDRESS, 0x0105)
4598 000026BB 66B80501    <1>    mov     ax, 0105h
4599 000026BF 66BACE03    <1>    mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
4600 000026C3 66EF    <1>    out     dx, ax
4601    <1>
4602    <1>    ; 07/08/2016
4603    <1>    ;mov     ah, [esp+1]
4604    <1>    ;movzx   edx, ah ; ysrc
4605 000026C5 0FB6542401    <1>    movzx   edx, byte [esp+1]
4606    <1>    ; 08/08/2016
4607 000026CA 0FB605[22680000]    <1>    movzx   eax, byte [CHAR_HEIGHT]
4608 000026D1 8A25[20680000]    <1>    mov     ah, [CRT_COLS] ; nbcols
4609 000026D7 F6E4    <1>    mul     ah
4610    <1>    ; 07/08/2016
4611    <1>    ;movzx   eax, byte [CRT_COLS]
4612    <1>    ;shl     ax, 3 ; * 8 ; * [CHAR_HEIGHT]
4613 000026D9 50    <1>    push    eax ; height * nbcols
4614 000026DA F7E2    <1>    mul     edx ; * ysrc
4615    <1>    ; eax = ysrc * height * nbcols
4616    <1>    ; edx = 0
4617 000026DC 88CA    <1>    mov     dl, cl ; edx = xstart
4618 000026DE 01D0    <1>    add     eax, edx
4619 000026E0 89C6    <1>    mov     esi, eax ; src
4620 000026E2 88EA    <1>    mov     dl, ch ; ydest
4621 000026E4 58    <1>    pop     eax ; height * nbcols
4622 000026E5 F7E2    <1>    mul     edx
4623    <1>    ; eax = ydest * height * nbcols
4624 000026E7 88CA    <1>    mov     dl, cl ; edx = xstart
4625 000026E9 01D0    <1>    add     eax, edx
4626 000026EB 89C7    <1>    mov     edi, eax ; dest
4627    <1>    ; esi = src
4628    <1>    ; edi = dest
4629    <1>    ; for(i=0;i<height;i++)
4630    <1>    ; {
4631    <1>    ;     memcpyb(0xa000,dest+i*nbcols,0xa000,src+i*nbcols,cols);
4632    <1>    ; }
4633 000026ED 51    <1>    push    ecx
4634 000026EE B9000000A00    <1>    mov     ecx, 0A0000h
4635 000026F3 01CE    <1>    add     esi, ecx
4636 000026F5 01CF    <1>    add     edi, ecx
4637    <1>    ; 08/08/2016

```

```

4638 000026F7 8A35[22680000] <1> mov dh, [CHAR_HEIGHT]
4639 <1> ;; 07/08/2016
4640 <1> ;mov dh, 8 ; 07/08/2016
4641 000026FD 28D2 <1> sub dl, dl ; i
4642 <1> vgamem_copy_p14_0:
4643 000026FF 56 <1> push esi
4644 00002700 57 <1> push edi
4645 00002701 0FB605[20680000] <1> movzx eax, byte [CRT_COLS]
4646 00002708 F6E2 <1> mul dl
4647 <1> ; eax = i * nbcols
4648 0000270A 01C7 <1> add edi, eax ; dest+i*nbcols
4649 0000270C 01C6 <1> add esi, eax
4650 0000270E 0FB6CF <1> movzx ecx, bh ; cols
4651 00002711 F3A4 <1> rep movsb
4652 00002713 5F <1> pop edi
4653 00002714 5E <1> pop esi
4654 00002715 FECE <1> dec dh
4655 00002717 75E6 <1> jnz short vgamem_copy_p14_0
4656 <1> vgamem_copy_p14_1:
4657 00002719 59 <1> pop ecx
4658 <1>
4659 <1> ; outw(VGAREG_GRDC_ADDRESS, 0x0005);
4660 0000271A 66B80500 <1> mov ax, 0005h
4661 0000271E 66BACE03 <1> mov dx, 3CEh ; VGAREG_GRDC_ADDRESS
4662 00002722 66EF <1> out dx, ax
4663 <1>
4664 00002724 58 <1> pop eax
4665 00002725 5A <1> pop edx
4666 <1>
4667 00002726 C3 <1> retn
4668 <1>
4669 <1> vgamem_fill_p14:
4670 <1> ; 08/08/2016
4671 <1> ; 07/08/2016
4672 <1> ; 04/08/2016
4673 <1> ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
4674 <1> ;
4675 <1> ; derived from 'Plex86/Bochs VGABios' source code
4676 <1> ; vgabios-0.7a (2011)
4677 <1> ; by the LGPL VGABios developers Team (2001-2008)
4678 <1> ; 'vgabios.c', 'vgamem_fill_p14'
4679 <1> ;
4680 <1> ; vgamem_fill_p14(xstart, ystart, cols, nbcols, cheight, attr)
4681 <1> ; cl = xstart, edi = ch = ystart, bh = cols,
4682 <1> ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight, attr = 0
4683 <1>
4684 <1> ; dest=ystart*cheight*nbcols+xstart;
4685 00002727 52 <1> push edx
4686 00002728 50 <1> push eax
4687 <1>
4688 <1> ; outw(VGAREG_GRDC_ADDRESS, 0x0205)
4689 00002729 66B80502 <1> mov ax, 0205h
4690 0000272D 66BACE03 <1> mov dx, 3CEh ; VGAREG_GRDC_ADDRESS
4691 00002731 66EF <1> out dx, ax
4692 <1>
4693 <1> ; 08/08/2016
4694 00002733 0FB605[22680000] <1> movzx eax, byte [CHAR_HEIGHT]
4695 0000273A F6E5 <1> mul ch
4696 <1> ;; 07/08/2016
4697 <1> ;movzx eax, ch
4698 <1> ;shl ax, 3 ; * 8 ; * [CHAR_HEIGHT]
4699 0000273C 0FB615[20680000] <1> movzx edx, byte [CRT_COLS] ; = [VGA_COLS]
4700 00002743 F7E2 <1> mul edx
4701 <1> ; edx = 0
4702 00002745 88CA <1> mov dl, cl
4703 00002747 01D0 <1> add eax, edx
4704 00002749 89C7 <1> mov edi, eax
4705 <1> ; edi = dest
4706 <1> ; for(i=0;i<cheight;i++)
4707 <1> ; {
4708 <1> ; memsetw(0xa000, dest+i*nbcols, attr, cols);
4709 <1> ; }
4710 0000274B 81C70000A00 <1> add edi, 0A0000h
4711 00002751 51 <1> push ecx
4712 <1> ; 08/08/2016
4713 00002752 8A35[22680000] <1> mov dh, [CHAR_HEIGHT]
4714 <1> ;; 07/08/2016
4715 <1> ;mov dh, 8 ; 07/08/2016
4716 00002758 28D2 <1> sub dl, dl ; i
4717 <1> vgamem_fill_p14_0:
4718 0000275A 57 <1> push edi
4719 0000275B 0FB605[20680000] <1> movzx eax, byte [CRT_COLS]
4720 00002762 F6E2 <1> mul dl
4721 <1> ; eax = i * nbcols
4722 00002764 01C7 <1> add edi, eax ; dest+i*nbcols
4723 00002766 88D8 <1> mov al, bl ; attr ; 04/08/2016
4724 00002768 0FB6CF <1> movzx ecx, bh ; cols
4725 0000276B F3AA <1> rep stosb
4726 0000276D 5F <1> pop edi
4727 0000276E 75EA <1> jnz short vgamem_fill_p14_0
4728 <1> vgamem_fill_p14_1:
4729 00002770 59 <1> pop ecx
4730 <1>
4731 <1> ; outw(VGAREG_GRDC_ADDRESS, 0x0005);
4732 00002771 66B80500 <1> mov ax, 0005h
4733 00002775 66BACE03 <1> mov dx, 3CEh ; VGAREG_GRDC_ADDRESS
4734 00002779 66EF <1> out dx, ax
4735 <1>
4736 0000277B 58 <1> pop eax
4737 0000277C 5A <1> pop edx
4738 <1>
4739 0000277D C3 <1> retn
4740 <1>
4741 <1> vgamem_copy_l8:
4742 <1> ; 02/08/2022 - TRDOS 386 kernel v2.0.5
4743 <1> ; 08/08/2016
4744 <1> ; 07/08/2016
4745 <1> ; 06/08/2016
4746 <1> ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
4747 <1> ;
4748 <1> ; TEMPORARY
4749 <1> ;
4750 <1> ; derived from 'Plex86/Bochs VGABios' source code
4751 <1> ; vgabios-0.7a (2011)
4752 <1> ; by the LGPL VGABios developers Team (2001-2008)
4753 <1> ; 'vgabios.c', 'vgamem_copy_p14'
4754 <1> ;
4755 <1> ; vgamem_copy_p14(xstart, ysrc, ydest, cols, nbcols, cheight)
4756 <1> ; cl = xstart, ah = ysrc (i+nblines), ch = ydest (i),
4757 <1> ; bh = cols, [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight
4758 <1>
4759 <1> ; src=ysrc*cheight*nbcols+xstart;
4760 <1> ; dest=ydest*cheight*nbcols+xstart;
4761 <1>

```



```

4762 0000277E 52      <1>    push    edx
4763 0000277F 50      <1>    push    eax
4764                <1>
4765                <1>    ; outw(VGAREG_GRDC_ADDRESS, 0x0105)
4766                <1>    ;mov    ax, 0105h
4767                <1>    ;mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4768                <1>    ;out    dx, ax
4769                <1>
4770                <1>    ;mov    ah, [esp+1]
4771                <1>
4772 00002780 0FB6D4    <1>    movzx   edx, ah ; ysrc
4773                <1>    ; 08/08/2016
4774 00002783 0FB605[22680000] <1>    movzx   eax, byte [CHAR_HEIGHT]
4775 0000278A 8A25[20680000] <1>    mov     ah, [CRT_COLS] ; nbcols
4776 00002790 F6E4      <1>    mul     ah
4777                <1>    ; 07/08/2016
4778                <1>    ;movzx  eax, byte [CRT_COLS]
4779                <1>    ;shl    ax, 3 ; * 8 ; * [CHAR_HEIGHT]
4780 00002792 50      <1>    push    eax ; cheight * nbcols
4781 00002793 F7E2      <1>    mul     edx ; * ysrc
4782                <1>    ; eax = ysrc * cheight * nbcols
4783                <1>    ; edx = 0
4784 00002795 88CA      <1>    mov     dl, cl ; edx = xstart
4785 00002797 01D0      <1>    add     eax, edx
4786 00002799 89C6      <1>    mov     esi, eax ; src
4787                <1>    ;shl    si, 3 ; * 8 ; 06/08/2016
4788                <1>    ; 02/08/2022
4789 0000279B C1E603    <1>    shl     esi, 3
4790 0000279E 88EA      <1>    mov     dl, ch ; ydest
4791 000027A0 58      <1>    pop     eax ; cheight * nbcols
4792 000027A1 F7E2      <1>    mul     edx
4793                <1>    ; eax = ydest * cheight * nbcols
4794 000027A3 88CA      <1>    mov     dl, cl ; edx = xstart
4795 000027A5 01D0      <1>    add     eax, edx
4796 000027A7 89C7      <1>    mov     edi, eax ; dest
4797                <1>    ;shl    di, 3 ; * 8 ; 06/08/2016
4798                <1>    ; 02/08/2022
4799 000027A9 C1E703    <1>    shl     edi, 3
4800                <1>    ; esi = src
4801                <1>    ; edi = dest
4802                <1>    ; for(i=0;i<cheight;i++)
4803                <1>    ; {
4804                <1>    ;     memcpyb(0xa000,dest+i*nbcols,0xa000,src+i*nbcols,cols);
4805                <1>    ; }
4806 000027AC 51      <1>    push    ecx
4807 000027AD B900000A00    <1>    mov     ecx, 0A0000h
4808 000027B2 01CE      <1>    add     esi, ecx
4809 000027B4 01CF      <1>    add     edi, ecx
4810                <1>    ; 08/08/2016
4811 000027B6 8A35[22680000] <1>    mov     dh, [CHAR_HEIGHT]
4812                <1>    ; 07/08/2016
4813                <1>    ;mov    dh, 8 ; 07/08/2016
4814 000027BC 28D2      <1>    sub     dl, dl ; i
4815                <1>    vgamem_copy_l8_0:
4816 000027BE 56      <1>    push    esi
4817 000027BF 57      <1>    push    edi
4818 000027C0 0FB605[20680000] <1>    movzx   eax, byte [CRT_COLS]
4819 000027C7 F6E2      <1>    mul     dl
4820                <1>    ; eax = i * nbcols
4821                <1>    ;shl    ax, 3 ; * 8 ; 06/08/2016
4822                <1>    ; 02/08/2022
4823 000027C9 C1E003    <1>    shl     eax, 3
4824 000027CC 01C7      <1>    add     edi, eax ; dest+i*nbcols
4825 000027CE 01C6      <1>    add     esi, eax
4826 000027D0 0FB6CF      <1>    movzx   ecx, bh ; cols
4827                <1>    ;shl    cx, 3 ; * 8 ; 06/08/2016
4828                <1>    ; 02/08/2022
4829 000027D3 C1E103    <1>    shl     ecx, 3
4830 000027D6 F3A4      <1>    rep     movsb
4831 000027D8 5F      <1>    pop     edi
4832 000027D9 5E      <1>    pop     esi
4833 000027DA FEC2      <1>    inc     dl ; 06/08/2016
4834 000027DC FECE      <1>    dec     dh
4835 000027DE 75DE      <1>    jnz     short vgamem_copy_l8_0
4836                <1>    vgamem_copy_l8_1:
4837 000027E0 59      <1>    pop     ecx
4838                <1>
4839                <1>    ; outw(VGAREG_GRDC_ADDRESS, 0x0005);
4840                <1>    ;mov    ax, 0005h
4841                <1>    ;mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4842                <1>    ;out    dx, ax
4843                <1>
4844 000027E1 58      <1>    pop     eax
4845 000027E2 5A      <1>    pop     edx
4846                <1>
4847 000027E3 C3      <1>    retn
4848                <1>
4849                <1>    vgamem_fill_l8:
4850                <1>    ; 02/08/2022 - TRDOS 386 kernel v2.0.5
4851                <1>    ; 08/08/2016
4852                <1>    ; 07/08/2016
4853                <1>    ; 06/08/2016
4854                <1>    ; 04/08/2016
4855                <1>    ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
4856                <1>    ;
4857                <1>    ; TEMPORARY
4858                <1>    ;
4859                <1>    ; derived from 'Plex86/Bochs VGABios' source code
4860                <1>    ; vgabios-0.7a (2011)
4861                <1>    ; by the LGPL VGABios developers Team (2001-2008)
4862                <1>    ; 'vgabios.c', 'vgamem_fill_p14'
4863                <1>    ;
4864                <1>    ; vgamem_fill_p14(xstart,ystart,cols,nbcols,cheight,attr)
4865                <1>    ; cl = xstart, edi = ch = ystart, bh = cols,
4866                <1>    ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = cheight, attr = 0
4867                <1>
4868                <1>    ; dest=ystart*cheight*nbcols+xstart;
4869 000027E4 52      <1>    push    edx
4870 000027E5 50      <1>    push    eax
4871                <1>
4872                <1>    ; outw(VGAREG_GRDC_ADDRESS, 0x0205)
4873                <1>    ;mov    ax, 0205h
4874                <1>    ;mov    dx, 3CEh ; VGAREG_GRDC_ADDRESS
4875                <1>    ;out    dx, ax
4876                <1>
4877                <1>    ; 08/08/2016
4878 000027E6 0FB605[22680000] <1>    movzx   eax, byte [CHAR_HEIGHT]
4879 000027ED F6E5      <1>    mul     ch
4880                <1>    ; 07/08/2016
4881                <1>    ;movzx  eax, ch
4882                <1>    ;shl    ax, 3 ; * 8 ; * [CHAR_HEIGHT]
4883 000027EF 0FB615[20680000] <1>    movzx   edx, byte [CRT_COLS] ; = [VGA_COLS]
4884 000027F6 F7E2      <1>    mul     edx
4885                <1>    ; edx = 0

```

```

4886 000027F8 88CA      <1>      mov     dl, cl
4887 000027FA 01D0      <1>      add     eax, edx
4888 000027FC 89C7      <1>      mov     edi, eax
4889                      <1>      ;shl     dl, 3 ; * 8 ; 06/08/2016
4890                      <1>      ; 02/08/2022
4891 000027FE C1E703    <1>      shl     edi, 3
4892                      <1>      ; edi = dest
4893                      <1>      ; for(i=0;i<cheight;i++)
4894                      <1>      ; {
4895                      <1>      ;   memsetb(0xa000,dest+i*nbcols,attr,cols);
4896                      <1>      ; }
4897 00002801 81C70000A00    <1>      add     edi, 0A0000h
4898 00002807 51          <1>      push    ecx
4899                      <1>      ; 08/08/2016
4900 00002808 8A35[22680000] <1>      mov     dh, [CHAR_HEIGHT]
4901                      <1>      ; 07/08/2016
4902                      <1>      ;mov     dh, 8 ; 07/08/2016
4903 0000280E 28D2      <1>      sub     dl, dl ; i
4904                      <1>      vgamem_fill_l8_0:
4905 00002810 57          <1>      push    edi
4906 00002811 0FB605[20680000] <1>      movzx   eax, byte [CRT_COLS]
4907 00002818 F6E2      <1>      mul     dl
4908                      <1>      ; eax = i * nbcols
4909                      <1>      ;shl     ax, 3 ; * 8 ; 06/08/2016
4910                      <1>      ; 02/08/2022
4911 0000281A C1E003    <1>      shl     eax, 3
4912 0000281D 01C7      <1>      add     edi, eax ; dest+i*nbcols
4913 0000281F 88D8      <1>      mov     al, bl ; attr ; 04/08/2016
4914 00002821 0FB6CF      <1>      movzx   ecx, bh ; cols
4915                      <1>      ;shl     cx, 3 ; * 8 ; 06/08/2016
4916                      <1>      ; 02/08/2022
4917 00002824 C1E103    <1>      shl     ecx, 3
4918 00002827 F3AA      <1>      rep     stosb
4919 00002829 5F          <1>      pop     edi
4920 0000282A FEC2      <1>      inc     dl ; 06/08/2016
4921 0000282C FECE      <1>      dec     dh
4922 0000282E 75E0      <1>      jnz     short vgamem_fill_l8_0
4923                      <1>      vgamem_fill_l8_1:
4924 00002830 59          <1>      pop     ecx
4925                      <1>
4926                      <1>      ; outw(VGAREG_GRDC_ADDRESS, 0x0005);
4927                      <1>      ;mov     ax, 0005h
4928                      <1>      ;mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
4929                      <1>      ;out     dx, ax
4930                      <1>
4931 00002831 58          <1>      pop     eax
4932 00002832 5A          <1>      pop     edx
4933                      <1>
4934 00002833 C3          <1>      ret     n
4935                      <1>
4936                      <1>      ; 03/08/2022 - TRDOS 386 Kernel v2.0.5
4937                      <1>      ; 07/07/2016
4938                      <1>      ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
4939                      <1>      ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
4940                      <1>      ;-----
4941                      <1>      ; SCROLL DOWN
4942                      <1>      ; THIS ROUTINE SCROLLS DOWN THE INFORMATION ON THE CRT
4943                      <1>      ; ENTRY --
4944                      <1>      ; CH,CL = UPPER LEFT CORNER OF REGION TO SCROLL
4945                      <1>      ; DH,DL = LOWER RIGHT CORNER OF REGION TO SCROLL
4946                      <1>      ; BOTH OF THE ABOVE ARE IN CHARACTER POSITIONS
4947                      <1>      ; BH = FILL VALUE FOR BLANKED LINES
4948                      <1>      ; AL = # LINES TO SCROLL (AL=0 MEANS BLANK THE ENTIRE FIELD)
4949                      <1>      ; DS = DATA SEGMENT
4950                      <1>      ; ES = REGEN SEGMENT
4951                      <1>      ; EXIT --
4952                      <1>      ; NOTHING, THE SCREEN IS SCROLLED
4953                      <1>      ;-----
4954                      <1>
4955                      <1>      ; cl = upper left column
4956                      <1>      ; ch = upper left row
4957                      <1>      ; dl = lower right column
4958                      <1>      ; dh = lower right row
4959                      <1>      ;
4960                      <1>      ; al = line count (AL=0 means blank entire fields)
4961                      <1>      ; bl = fill value for blanked lines
4962                      <1>      ; bh = unused
4963                      <1>
4964                      <1>      GRAPHICS_DOWN:
4965                      <1>      ; 07/07/2016
4966                      <1>      ; ah = Current video mode, [CRT_MODE]
4967                      <1>      ; std     ; SET DIRECTION
4968 00002834 80FC07    <1>      cmp     ah, 7
4969 00002837 7769      <1>      ja     short vga_graphics_down ; 03/08/2022
4970                      <1>      ;je     _n0
4971                      <1>
4972 00002839 88C7      <1>      mov     bh, al ; save line count in BH
4973                      <1>      ;mov     ax, dx ; GET LOWER RIGHT POSITION INTO AX REG
4974                      <1>      ; 03/08/2022
4975 0000283B 89D0      <1>      mov     eax, edx
4976                      <1>
4977                      <1>      ;----- USE CHARACTER SUBROUTINE FOR POSITIONING
4978                      <1>      ;----- ADDRESS RETURNED IS MULTIPLIED BY 2 FROM CORRECT VALUE
4979                      <1>
4980 0000283D E8E8020000 <1>      call    GRAPH_POSN
4981                      <1>      ;movzx   edi, ax ; SAVE RESULT AS DESTINATION ADDRESS
4982                      <1>      ; 03/08/2022
4983 00002842 89C7      <1>      mov     edi, eax
4984                      <1>
4985                      <1>      ;----- DETERMINE SIZE OF WINDOW
4986                      <1>
4987 00002844 6629CA      <1>      sub     dx, cx
4988 00002847 6681C20101 <1>      add     dx, 101h ; ADJUST VALUES
4989 0000284C C0E602      <1>      sal     dh, 2 ; MULTIPLY ROWS BY 4 AT 8 VERT DOTS/CHAR
4990                      <1>      ; AND EVEN/ODD ROWS
4991                      <1>      ;----- DETERMINE CRT MODE
4992                      <1>
4993 0000284F 803D[1E680000]06 <1>      cmp     byte [CRT_MODE], 6 ; TEST FOR MEDIUM RES
4994 00002856 7305      <1>      jnc     short _R12 ; FIND_SOURCE_DOWN
4995                      <1>
4996                      <1>      ;----- MEDIUM RES DOWN
4997 00002858 D0E2      <1>      sal     dl, 1 ; # COLUMNS * 2, SINCE 2 BYTES/CHAR
4998                      <1>      ;sal     di, 1 ; OFFSET *2 SINCE 2 BYTES/CHAR
4999                      <1>      ; 03/08/2022
5000 0000285A D1E7      <1>      sal     edi, 1
5001                      <1>      ;inc     di ; POINT TO LAST BYTE
5002                      <1>      ; 03/08/2022
5003 0000285C 47          <1>      inc     edi
5004                      <1>
5005                      <1>      ;----- DETERMINE THE SOURCE ADDRESS IN THE BUFFER
5006                      <1>
5007                      <1>      _R12: ; FIND_SOURCE_DOWN
5008 0000285D 81C700800B00 <1>      add     edi, 0B8000h
5009 00002863 6681C7F000 <1>      add     di, 240 ; POINT TO LAST ROW OF PIXELS

```

```

5010 00002868 C0E702      <1>      sal      bh, 2          ; multiply number of lines by 4
5011 00002868 74(06)     <1>      jz       short 6        ; IF ZERO, THEN BLANK ENTIRE FIELD
5012 0000286D B050      <1>      mov      al, 80          ; 80 BYTES/ROW
5013 0000286F F6E7      <1>      mul      bh          ; determine offset to source
5014 00002871 89FE      <1>      mov      esi, edi       ; SET UP SOURCE
5015          <1>      ;sub      si, ax          ; SUBTRACT THE OFFSET
5016          <1>      ; 03/08/2022
5017 00002873 29C6      <1>      sub      esi, eax
5018 00002875 88F4      <1>      mov      ah, dh          ; NUMBER OF ROWS IN FIELD
5019 00002877 28FC      <1>      sub      ah, bh          ; determine number to move
5020          <1>
5021          <1>      ;-----      LOOP THROUGH, MOVING ONE ROW AT A TIME, BOTH EVEN AND ODD FIELDS
5022          <1>
5023          <1>      _R13:          ; ROW_LOOP_DOWN
5024 00002879 E825010000    <1>      call     _R17          ; MOVE ONE ROW
5025 0000287E 6681EE5020    <1>      sub      si, 2000h+80      ; MOVE TO NEXT ROW
5026 00002883 6681EF5020    <1>      sub      di, 2000h+80
5027 00002888 FECC      <1>      dec      ah          ; NUMBER OF ROWS TO MOVE
5028 0000288A 75ED      <1>      jnz      short _R13      ; CONTINUE TILL ALL MOVED
5029          <1>
5030          <1>      ;-----      FILL IN THE VACATED LINE(S)
5031          <1>      _R14:          ; CLEAR_ENTRY_DOWN
5032 0000288C 88D8      <1>      mov      al, b1          ; attribute to fill with
5033          <1>      _R15:          ; CLEAR_LOOP_DOWN
5034 0000288E E82C010000    <1>      call     _R18          ; CLEAR A ROW
5035 00002893 6681EF5020    <1>      sub      di, 2000h+80      ; POINT TO NEXT LINE
5036 00002898 FECF      <1>      dec      bh          ; number of lines to fill
5037 0000289A 75F2      <1>      jnz      short _R15_      ; CLEAR_LOOP_DOWN
5038          <1>      ; 18/11/2020
5039 0000289C FC          <1>      cld          ; RESET THE DIRECTION FLAG
5040          <1>
5041 0000289D C3          <1>      retn          ; EVERYTHING DONE
5042          <1>
5043          <1>      _R16:          ; BLANK_FIELD_DOWN
5044 0000289E 88F7      <1>      mov      bh, dh          ; set blank count to everything in field
5045 000028A0 EBEA      <1>      jmp      short _R14      ; CLEAR THE FIELD
5046          <1>
5047          <1>      vga_graphics_down:
5048          <1>      ; 03/08/2022 - TRDOS 386 Kertnel v2.0.5
5049          <1>      ; 12/04/2021
5050          <1>      ; 08/08/2016
5051          <1>      ; 07/08/2016
5052          <1>      ; 31/07/2016
5053          <1>      ; 07/07/2016 - TRDOS 386 (TRDOS v2.0)
5054          <1>      ;
5055          <1>      ; derived from 'Plex86/Bochs VGABios' source code
5056          <1>      ; vgabios-0.7a (2011)
5057          <1>      ; by the LGPL VGABios developers Team (2001-2008)
5058          <1>      ; 'vgabios.c', 'biosfn_scroll'
5059          <1>      ;
5060          <1>      ;
5061          <1>      ; cl = upper left column
5062          <1>      ; ch = upper left row
5063          <1>      ; dl = lower righth column
5064          <1>      ; dh = lower right row
5065          <1>      ;
5066          <1>      ; al = line count (AL=0 means blank entire fields)
5067          <1>      ; bl = fill value for blanked lines
5068          <1>      ; bh = unused
5069          <1>      ;
5070          <1>      ; ah = [CRT_MODE], current video mode
5071          <1>
5072 000028A2 FC          <1>      cld          ; !!! Clear direction flag !!!
5073          <1>
5074 000028A3 88C7      <1>      mov      bh, al ; 31/07/2016
5075          <1>
5076 000028A5 BE[3A680000] <1>      mov      esi, vga_modes
5077 000028AA 89F7      <1>      mov      edi, esi
5078 000028AC 83C710     <1>      add      edi, vga_mode_count
5079          <1>      vga_g_down_0:
5080 000028AF AC          <1>      lodsb
5081 000028B0 38E0      <1>      cmp      al, ah ; [CRT_MODE]
5082 000028B2 7405      <1>      je       short vga_g_down_1
5083 000028B4 39FE      <1>      cmp      esi, edi
5084 000028B6 72F7      <1>      jb       short vga_g_down_0
5085          <1>      ; xor      bh, bh ; 31/07/2016
5086 000028B8 C3          <1>      retn          ; nothing to do
5087          <1>      vga_g_down_1:
5088 000028B9 88F8      <1>      mov      al, bh ; 31/07/2016
5089 000028BB 83C64F     <1>      add      esi, vga_memmodel - (vga_modes + 1)
5090          <1>      ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
5091          <1>
5092          <1>      ; if(rlr>nbrows)rlr=nbrows-1;
5093          <1>      ; if(clr>nbcols)clr=nbcols-1;
5094          <1>      ; if(nblines>nbrows)nblines=0;
5095          <1>      ; cols=clr-cul+1;
5096          <1>
5097 000028BE 3A35[26680000] <1>      cmp      dh, [VGA_ROWS]
5098 000028C4 7208      <1>      jb       short vga_g_down_2
5099 000028C6 8A35[26680000] <1>      mov      dh, [VGA_ROWS]
5100 000028CC FECE      <1>      dec      dh
5101          <1>      vga_g_down_2:
5102 000028CE 3A15[20680000] <1>      cmp      dl, [CRT_COLS] ; = [VGA_COLS]
5103 000028D4 7208      <1>      jb       short vga_g_down_3
5104 000028D6 8A15[20680000] <1>      mov      dl, [CRT_COLS]
5105 000028DC FECA      <1>      dec      dl
5106          <1>      vga_g_down_3:
5107 000028DE 3A05[26680000] <1>      cmp      al, [VGA_ROWS]
5108 000028E4 7602      <1>      jna       short vga_g_down_4
5109 000028E6 28C0      <1>      sub      al, al ; 0
5110          <1>      vga_g_down_4:
5111 000028E8 88F7      <1>      mov      bh, dh ; clr
5112 000028EA 28CF      <1>      sub      bh, cl ; cul
5113 000028EC FEC7      <1>      inc      bh ; cols = clr-cul+1
5114          <1>
5115 000028EE 20C0      <1>      and      al, al ; nblines = 0
5116 000028F0 7559      <1>      jnz      short vga_g_down_6
5117 000028F2 20ED      <1>      and      ch, ch ; rul = 0
5118 000028F4 7555      <1>      jnz      short vga_g_down_6
5119 000028F6 20C9      <1>      and      cl, cl ; cul = 0
5120 000028F8 7551      <1>      jnz      short vga_g_down_6
5121          <1>
5122 000028FA 50          <1>      push     eax ; push ax ; 12/04/2021
5123 000028FB A0[26680000] <1>      mov      al, [VGA_ROWS]
5124 00002900 FEC8      <1>      dec      al
5125 00002902 38C6      <1>      cmp      dh, al ; rlr = nbrows-1
5126 00002904 7545      <1>      jne       short vga_g_down_5
5127 00002906 A0[20680000] <1>      mov      al, [CRT_COLS] ; = VGA_COLS
5128 0000290B FEC8      <1>      dec      al
5129 0000290D 38C2      <1>      cmp      dl, al ; clr = nbcols-1
5130          <1>      ;jne       short vga_g_down_5
5131          <1>      ; 12/04/2021
5132 0000290F 58          <1>      pop      eax ; pop ax
5133 00002910 7539      <1>      jne       short vga_g_down_5

```

```

5134      <1>
5135 00002912 66B80502      <1>      mov     ax, 0205h
5136 00002916 66BACE03      <1>      mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
5137 0000291A 66EF          <1>      out     dx, ax
5138 0000291C A0[26680000]      <1>      mov     al, [VGA_ROWS]
5139 00002921 8A25[20680000]      <1>      mov     ah, [CRT_COLS] ; = [VGA_COLS]
5140 00002927 F6E4          <1>      mul     ah
5141 00002929 0FB7D0          <1>      movzx   edx, ax
5142          <1>      ; 08/08/2016
5143 0000292C 0FB605[22680000]      <1>      movzx   eax, byte [CHAR_HEIGHT]
5144 00002933 F7E2          <1>      mul     edx
5145          <1>      ; eax = byte count
5146 00002935 89C1          <1>      mov     ecx, eax
5147          <1>      ; 07/08/2016
5148          <1>      ; shl     dx, 3 ; * 8 ; * [CHAR_HEIGHT]
5149          <1>      ; mov     ecx, edx
5150 00002937 88D8          <1>      mov     al, bl ; fill value for blanked lines
5151 00002939 BF0000A00      <1>      mov     edi, 0A0000h
5152 0000293E F3AA          <1>      rep     stosb
5153          <1>
5154          <1>      ; 03/08/2022
5155 00002940 30E4          <1>      xor     ah, ah ; 0
5156          <1>
5157 00002942 B005          <1>      mov     al, 5
5158 00002944 66BACE03      <1>      mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
5159 00002948 66EF          <1>      out     dx, ax ; 0005h
5160          <1>
5161 0000294A C3          <1>      retn
5162          <1>
5163          <1> vga_g_down_5:
5164          <1>      ; 12/04/2021
5165          <1>      ; pop     eax ; pop ax
5166          <1>
5167          <1> vga_g_down_6:
5168          <1>      ; [ESI] = VGA memory model number for current video mode
5169          <1>      ;
5170          <1>      ; LINEAR8 equ 5
5171          <1>      ; PLANAR4 equ 4
5172          <1>      ; PLANAR1 equ 3
5173          <1>
5174 0000294B 803E04          <1>      cmp     byte [esi], PLANAR4
5175 0000294E 742C          <1>      je      short vga_g_down_planar
5176 00002950 803E03          <1>      cmp     byte [esi], PLANAR1
5177 00002953 7427          <1>      je      short vga_g_down_planar
5178          <1> vga_g_down_linear8:
5179          <1>      ; 07/07/2016 (TEMPORARY)
5180          <1>      ;
5181          <1>      ; cl = upper left column ; cul
5182          <1>      ; ch = upper left row ; rul
5183          <1>      ; dl = lower right column ; clr
5184          <1>      ; dh = lower right row ; rlr
5185          <1>
5186          <1> vga_g_down_l0:
5187          <1>      ; {for(i=rlr;i>=rul;i--)
5188          <1>      ; ; if((i<rul+nblines)|| (nblines==0))
5189          <1>      or      al, al
5190 00002955 08C0          <1>      jz      short vga_g_down_l2
5191 00002957 741C          <1>      mov     ah, al
5192 00002959 88C4          <1>      add     ah, ch
5193          <1>      ; jc      short vga_g_down_l2
5194 0000295D 86F5          <1>      xchg    ch, dh
5195 0000295F 38E5          <1>      cmp     ch, ah
5196 00002961 7212          <1>      jb      short vga_g_down_l2
5197 00002963 88EC          <1>      mov     ah, ch
5198 00002965 28C4          <1>      sub     ah, al ; ah = i - nblines
5199          <1>      ; else
5200          <1>      ; vgamem_copy_p14(cul,i,i-nblines,cols,nbcols,height);
5201 00002967 E812FEFFFF      <1>      call    vgamem_copy_l8
5202          <1> vga_g_down_l1:
5203          <1>      xchg    dh, ch
5204 0000296C 86EE          <1>      dec     dh
5205 0000296E FECE          <1>      cmp     dh, ch
5206 00002970 38EE          <1>      jnb     short vga_g_down_l0
5207 00002972 73E1          <1>
5208 00002974 C3          <1>      retn
5209          <1>
5210          <1> vga_g_down_l2:
5211          <1>      ; vgamem_fill_p14(cul,i,cols,nbcols,height,attr);
5212 00002975 E86AFEFFFF      <1>      call    vgamem_fill_l8
5213 0000297A EBF0          <1>      jmp     short vga_g_down_l1
5214          <1>
5215          <1> vga_g_down_planar:
5216          <1>      ; cl = upper left column ; cul
5217          <1>      ; ch = upper left row ; rul
5218          <1>      ; dl = lower right column ; clr
5219          <1>      ; dh = lower right row ; rlr
5220          <1> vga_g_down_p10:
5221          <1>      ; {for(i=rlr;i>=rul;i--)
5222          <1>      ; ; if((i<rul+nblines)|| (nblines==0))
5223          <1>      or      al, al
5224 0000297C 08C0          <1>      jz      short vga_g_down_p12
5225 0000297E 741C          <1>      mov     ah, al
5226 00002980 88C4          <1>      add     ah, ch
5227          <1>      ; jc      short vga_g_down_p12
5228 00002984 86F5          <1>      xchg    ch, dh
5229 00002986 38E5          <1>      cmp     ch, ah
5230 00002988 7212          <1>      jb      short vga_g_down_p12
5231 0000298A 88EC          <1>      mov     ah, ch
5232 0000298C 28C4          <1>      sub     ah, al ; ah = i - nblines
5233          <1>      ; else
5234          <1>      ; vgamem_copy_p14(cul,i,i-nblines,cols,nbcols,height);
5235 0000298E E826FDFFFF      <1>      call    vgamem_copy_p14
5236          <1> vga_g_down_p11:
5237          <1>      xchg    dh, ch
5238 00002993 86EE          <1>      dec     dh
5239 00002995 FECE          <1>      cmp     dh, ch
5240 00002997 38EE          <1>      jnb     short vga_g_down_p10
5241 00002999 73E1          <1>
5242 0000299B C3          <1>      retn
5243          <1>
5244          <1> vga_g_down_p12:
5245          <1>      ; vgamem_fill_p14(cul,i,cols,nbcols,height,attr);
5246 0000299C E886FDFFFF      <1>      call    vgamem_fill_p14
5247 000029A1 EBF0          <1>      jmp     short vga_g_down_p11
5248          <1>
5249          <1> ; 27/06/2016 - TRDOS 386 (TRDOS v2.0)
5250          <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5251          <1>
5252          <1> ;----- ROUTINE TO MOVE ONE ROW OF INFORMATION
5253          <1>
5254          <1> _R17:
5255 000029A3 0FB6CA          <1>      movzx   ecx, dl ; NUMBER OF BYTES IN THE ROW
5256 000029A6 56          <1>      push    esi
5257 000029A7 57          <1>      push    edi ; SAVE POINTERS

```

```

5258 000029A8 F3A4      <1>      rep      movsb          ; MOVE THE EVEN FIELD
5259 000029AA 5F        <1>      pop      edi
5260 000029AB 5E        <1>      pop      esi
5261 000029AC 6681C60020 <1>      add      si, 2000h
5262 000029B1 6681C70020 <1>      add      di, 2000h          ; POINT TO THE ODD FIELD
5263 000029B6 56        <1>      push     esi
5264 000029B7 57        <1>      push     edi          ; SAVE THE POINTERS
5265 000029B8 88D1      <1>      mov      cl, dl          ; COUNT BACK
5266 000029BA F3A4      <1>      rep      movsb          ; MOVE THE ODD FIELD
5267 000029BC 5F        <1>      pop      edi
5268 000029BD 5E        <1>      pop      esi          ; POINTERS BACK
5269 000029BE C3        <1>      retn          ; RETURN TO CALLER
5270
5271 <1>      ;-----      CLEAR A SINGLE ROW
5272 <1>
5273 <1>      _R18:
5274 000029BF 0FB6CA    <1>      movzx     ecx, dl          ; NUMBER OF BYTES IN FIELD
5275 000029C2 57        <1>      push     edi          ; SAVE POINTER
5276 000029C3 F3AA      <1>      rep      stosb          ; STORE THE NEW VALUE
5277 000029C5 5F        <1>      pop      edi          ; POINTER BACK
5278 000029C6 6681C70020 <1>      add      di, 2000h          ; POINT TO ODD FIELD
5279 000029CB 57        <1>      push     edi
5280 000029CC 88D1      <1>      mov      cl, dl
5281 000029CE F3AA      <1>      rep      stosb          ; FILL THE ODD FIELD
5282 000029D0 5F        <1>      pop      edi
5283 000029D1 C3        <1>      retn          ; RETURN TO CALLER
5284 <1>
5285 <1>      ; 03/08/2022 - TRDOS 386 kernel v2.0.5
5286 <1>      ; 04/07/2016
5287 <1>      ; 01/07/2016
5288 <1>      ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
5289 <1>      ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5290 <1>      ;-----
5291 <1>      ; GRAPHICS WRITE
5292 <1>      ; THIS ROUTINE WRITES THE ASCII CHARACTER TO THE CURRENT
5293 <1>      ; POSITION ON THE SCREEN.
5294 <1>      ; ENTRY --
5295 <1>      ; AL = CHARACTER TO WRITE
5296 <1>      ; BL = COLOR ATTRIBUTE TO BE USED FOR FOREGROUND COLOR
5297 <1>      ; IF BIT 7 IS SET, THE CHAR IS XOR'D INTO THE REGEN BUFFER
5298 <1>      ; (0 IS USED FOR THE BACKGROUND COLOR)
5299 <1>      ; CX = NUMBER OF CHARS TO WRITE
5300 <1>      ; DS = DATA SEGMENT
5301 <1>      ; ES = REGEN SEGMENT
5302 <1>      ; EXIT --
5303 <1>      ; NOTHING IS RETURNED
5304 <1>
5305 <1>      ; GRAPHICS READ
5306 <1>      ; THIS ROUTINE READS THE ASCII CHARACTER AT THE CURRENT CURSOR
5307 <1>      ; POSITION ON THE SCREEN BY MATCHING THE DOTS ON THE SCREEN TO THE
5308 <1>      ; CHARACTER GENERATOR CODE POINTS
5309 <1>      ; ENTRY --
5310 <1>      ; NONE (0 IS ASSUMED AS THE BACKGROUND COLOR)
5311 <1>      ; EXIT --
5312 <1>      ; AL = CHARACTER READ AT THAT POSITION (0 RETURNED IF NONE FOUND)
5313 <1>
5314 <1>      ; FOR BOTH ROUTINES, THE IMAGES USED TO FORM CHARS ARE CONTAINED IN ROM
5315 <1>      ; FOR THE 1ST 128 CHARS. TO ACCESS CHARS IN THE SECOND HALF, THE USER
5316 <1>      ; MUST INITIALIZE THE VECTOR AT INTERRUPT 1FH (LOCATION 0007CH) TO
5317 <1>      ; POINT TO THE USER SUPPLIED TABLE OF GRAPHIC IMAGES (8X8 BOXES).
5318 <1>      ; FAILURE TO DO SO WILL CAUSE IN STRANGE RESULTS
5319 <1>      ;-----
5320 <1>
5321 <1>      GRAPHICS_WRITE:
5322 000029D2 25FF000000 <1>      and      eax, 0FFh          ; ZERO TO HIGH OF CODE POINT
5323 000029D7 50        <1>      push     eax          ; SAVE CODE POINT VALUE
5324 <1>
5325 <1>      ;-----      DETERMINE POSITION IN REGEN BUFFER TO PUT CODE POINTS
5326 <1>
5327 000029D8 E846010000 <1>      call     S26          ; FIND LOCATION IN REGEN BUFFER
5328 000029DD 89C7      <1>      mov      edi, eax          ; REGEN POINTER IN DI
5329 <1>
5330 <1>      ;-----      DETERMINE REGION TO GET CODE POINTS FROM
5331 <1>
5332 000029DF 58        <1>      pop      eax          ; RECOVER CODE POINT
5333 <1>
5334 000029E0 BE[304D0100] <1>      mov      esi, CRT_CHAR_GEN ; OFFSET OF IMAGES
5335 <1>
5336 <1>      ;-----      DETERMINE GRAPHICS MODE IN OPERATION
5337 <1>      ; DETERMINE_MODE
5338 <1>      ; sal      ax, 3          ; MULTIPLY CODE POINT VALUE BY 8
5339 <1>      ; 03/08/2022
5340 000029E5 C1E003    <1>      sal      eax, 3
5341 000029E8 01C6      <1>      add      esi, eax          ; SI HAS OFFSET OF DESIRED CODES
5342 <1>
5343 000029EA 803D[1E680000]06 <1>      cmp      byte [CRT_MODE], 6
5344 000029F1 7231      <1>      jc       short S6          ; TEST FOR MEDIUM RESOLUTION MODE
5345 <1>
5346 <1>      ;-----      HIGH RESOLUTION MODE
5347 <1>
5348 000029F3 81C700800B00 <1>      add      edi, 0B8000h
5349 <1>      S1:
5350 000029F9 57        <1>      push     edi          ; HIGH_CHAR
5351 000029FA 56        <1>      push     esi          ; SAVE REGEN POINTER
5352 000029FB B604      <1>      mov      dh, 4          ; SAVE CODE POINTER
5353 <1>      S2:
5354 000029FD AC        <1>      lodsb          ; GET BYTE FROM CODE POINTS
5355 000029FE F6C380    <1>      test     bl, 80h          ; SHOULD WE USE THE FUNCTION
5356 00002A01 7515      <1>      jnz      short S5          ; TO PUT CHAR IN
5357 00002A03 AA        <1>      stosb          ; STORE IN REGEN BUFFER
5358 00002A04 AC        <1>      lodsb
5359 <1>      S4:
5360 00002A05 8887FF1F0000 <1>      mov      [edi+2000h-1], al ; STORE IN SECOND HALF
5361 00002A0B 83C74F    <1>      add      edi, 79          ; MOVE TO NEXT ROW IN REGEN
5362 00002A0E FECE      <1>      dec      dh          ; DONE WITH LOOP
5363 00002A10 75EB      <1>      jnz      short S2
5364 00002A12 5E        <1>      pop      esi
5365 00002A13 5F        <1>      pop      edi          ; RECOVER REGEN POINTER
5366 00002A14 47        <1>      inc      edi          ; POINT TO NEXT CHAR POSITION
5367 00002A15 E2E2      <1>      loop     S1          ; MORE CHARS TO WRITE
5368 00002A17 C3        <1>      retn
5369 <1>
5370 <1>      S5:
5371 00002A18 3207      <1>      xor      al, [edi]          ; EXCLUSIVE OR WITH CURRENT
5372 00002A1A AA        <1>      stosb          ; STORE THE CODE POINT
5373 00002A1B AC        <1>      lodsb          ; AGAIN FOR ODD FIELD
5374 00002A1C 3287FF1F0000 <1>      xor      al, [edi+2000h-1]
5375 00002A22 EBE1      <1>      jmp      short S4          ; BACK TO MAINSTREAM
5376 <1>
5377 <1>      ;-----      MEDIUM RESOLUTION WRITE
5378 <1>      S6:
5379 00002A24 88DA      <1>      mov      dl, bl          ; MED_RES_WRITE
5380 <1>      ; 03/08/2022
5381 00002A26 D1E7      <1>      sal      edi, 1

```

```

5382      <1>      ;sal      di, 1          ; OFFSET*2 SINCE 2 BYTES/CHAR
5383      <1>
5384      00002A28 80E303      <1>      and      bl, 3          ; EXPAND BL TO FULL WORD OF COLOR
5385      00002A2B B055      <1>      mov      al, 055h      ; ISOLATE THE COLOR BITS ( LOW 2 BITS )
5386      00002A2D F6E3      <1>      mul      bl          ; GET BIT CONVERSION MULTIPLIER
5387      00002A2F 88C3      <1>      mov      bl, al          ; EXPAND 2 COLOR BITS TO 4 REPLICATIONS
5388      00002A31 88C7      <1>      mov      bh, al          ; PLACE BACK IN WORK REGISTER
5389      00002A33 81C700800B00 <1>      add      edi, 0B8000h      ; EXPAND TO 8 REPLICATIONS OF COLOR BITS
5390      <1>      S7:          ; MED_CHAR
5391      00002A39 57      <1>      push     edi          ; SAVE REGEN POINTER
5392      00002A3A 56      <1>      push     esi          ; SAVE THE CODE POINTER
5393      00002A3B B604      <1>      mov      dh, 4          ; NUMBER OF LOOPS
5394      <1>      S8:
5395      00002A3D AC      <1>      lodsb          ; GET CODE POINT
5396      00002A3E E8B1000000 <1>      call     S21          ; DOUBLE UP ALL THE BITS
5397      00002A43 6621D8      <1>      and      ax, bx          ; CONVERT TO FOREGROUND COLOR ( 0 BACK )
5398      00002A46 86C4      <1>      xchg     ah, al          ; SWAP HIGH/LOW BYTES FOR WORD MOVE
5399      00002A48 F6C280      <1>      test     dl, 80h      ; IS THIS XOR FUNCTION
5400      00002A4B 7403      <1>      jz       short S9      ; NO, STORE IT IN AS IS
5401      00002A4D 663307      <1>      xor      ax, [edi]      ; DO FUNCTION WITH LOW/HIGH
5402      <1>      S9:
5403      00002A50 668907      <1>      mov      [edi], ax      ; STORE FIRST BYTE HIGH, SECOND LOW
5404      00002A53 AC      <1>      lodsb          ; GET CODE POINT
5405      00002A54 E89B000000 <1>      call     S21          ; CONVERT TO COLOR
5406      00002A59 6621D8      <1>      and      ax, bx          ; SWAP HIGH/LOW BYTES FOR WORD MOVE
5407      00002A5C 86C4      <1>      xchg     ah, al          ; AGAIN, IS THIS XOR FUNCTION
5408      00002A5E F6C280      <1>      test     dl, 80h      ; NO, JUST STORE THE VALUES
5409      00002A61 7407      <1>      jz       short _S10      ; FUNCTION WITH FIRST HALF LOW
5410      00002A63 66338700200000 <1>      xor      ax, [edi+2000h]
5411      <1>      _S10:
5412      00002A6A 66898700200000 <1>      mov      [edi+2000h], ax      ; STORE SECOND PORTION HIGH
5413      00002A71 6683C750      <1>      add      di, 80          ; POINT TO NEXT LOCATION
5414      00002A75 FECE      <1>      dec      dh
5415      00002A77 75C4      <1>      jnz      short S8          ; KEEP GOING
5416      00002A79 5E      <1>      pop      esi          ; RECOVER CODE POINTER
5417      00002A7A 5F      <1>      pop      edi          ; RECOVER REGEN POINTER
5418      00002A7B 47      <1>      inc      edi          ; POINT TO NEXT CHAR POSITION
5419      00002A7C 47      <1>      inc      edi
5420      00002A7D E2BA      <1>      loop     S7          ; MORE TO WRITE
5421      00002A7F C3      <1>      retn
5422      <1>
5423      <1>      ; 03/08/2022 - TRDOS 386 Kernel v2.0.5
5424      <1>      ; 04/07/2016
5425      <1>      ; 01/07/2016
5426      <1>      ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
5427      <1>      ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5428      <1>      ;-----
5429      <1>      ; GRAPHICS READ
5430      <1>      ;-----
5431      <1>      GRAPHICS_READ:
5432      00002A80 E89E000000      <1>      call     S26          ; CONVERTED TO OFFSET IN REGEN
5433      00002A85 89C6      <1>      mov      esi, eax          ; SAVE IN SI
5434      00002A87 81C600800B00 <1>      add      esi, 0B8000h      ; 01/07/2016
5435      00002A8D 83EC08      <1>      sub      esp, 8          ; ALLOCATE SPACE FOR THE READ CODE POINT
5436      00002A90 89E5      <1>      mov      ebp, esp          ; POINTER TO SAVE AREA
5437      <1>
5438      <1>      ;----- DETERMINE GRAPHICS MODES
5439      00002A92 B604      <1>      mov      dh, 4          ; number of passes ; 01/07/2016
5440      00002A94 803D[1E680000]06 <1>      cmp      byte [CRT_MODE], 6
5441      00002A9B 7219      <1>      jc       short S12          ; MEDIUM RESOLUTION
5442      <1>
5443      <1>      ;----- HIGH RESOLUTION READ
5444      <1>      ;----- GET VALUES FROM REGEN BUFFER AND CONVERT TO CODE POINT
5445      <1>      ;mov      dh,4          ; NUMBER OF PASSES
5446      <1>      S11:
5447      00002A9D 8A06      <1>      mov      al, [esi]          ; GET FIRST BYTE
5448      00002A9F 884500      <1>      mov      [ebp], al          ; SAVE IN STORAGE AREA
5449      00002AA2 45      <1>      inc      ebp          ; NEXT LOCATION
5450      00002AA3 8A8600200000 <1>      mov      al, [esi+2000h]      ; GET LOWER REGION BYTE
5451      00002AA9 884500      <1>      mov      [ebp], al          ; ADJUST AND STORE
5452      00002AAC 45      <1>      inc      ebp
5453      00002AAD 83C650      <1>      add      esi, 80          ; POINTER INTO REGEN
5454      00002AB0 FECE      <1>      dec      dh          ; LOOP CONTROL
5455      00002AB2 75E9      <1>      jnz      short S11          ; DO IT SOME MORE
5456      00002AB4 EB1C      <1>      jmp      short S14          ; GO MATCH THE SAVED CODE POINTS
5457      <1>
5458      <1>      ;----- MEDIUM RESOLUTION READ
5459      <1>      S12:
5460      <1>      ;sal      si, 1          ; OFFSET*2 SINCE 2 BYTES/CHAR
5461      <1>      ; 03/08/2022
5462      00002AB6 D1E6      <1>      sal      esi, 1
5463      <1>      ;mov      dh, 4          ; NUMBER OF PASSES
5464      <1>      S13:
5465      00002AB8 E84B000000      <1>      call     S23          ; GET BYTES FROM REGEN INTO SINGLE SAVE
5466      00002ABD 81C6FE1F0000 <1>      add      esi, 2000h-2      ; GO TO LOWER REGION
5467      00002AC3 E840000000      <1>      call     S23          ; GET THIS PAIR INTO SAVE
5468      00002AC8 81EEB21F0000 <1>      sub      esi, 2000h-80+2      ; ADJUST POINTER BACK INTO UPPER
5469      00002ACE FECE      <1>      dec      dh
5470      00002AD0 75E6      <1>      jnz      short S13          ; KEEP GOING UNTIL ALL 8 DONE
5471      <1>
5472      <1>      ;----- SAVE AREA HAS CHARACTER IN IT, MATCH IT
5473      <1>      S14:
5474      00002AD2 BF[304D0100] <1>      mov      edi, CRT_CHAR_GEN      ; FIND_CHAR
5475      00002AD7 83ED08      <1>      sub      ebp, 8          ; ESTABLISH ADDRESSING
5476      00002ADA 89EE      <1>      mov      esi, ebp          ; ADJUST POINTER TO START OF SAVE AREA
5477      <1>      S15:
5478      <1>      ;mov      ax, 256          ; NUMBER TO TEST AGAINST
5479      <1>      ; 03/08/2022
5480      00002ADC 29C0      <1>      sub      eax, eax
5481      00002ADE FEC4      <1>      inc      ah
5482      <1>      ; eax = 256
5483      <1>      S16:
5484      00002AE0 56      <1>      push     esi          ; SAVE SAVE AREA POINTER
5485      00002AE1 57      <1>      push     edi          ; SAVE CODE POINTER
5486      <1>      ;mov      ecx, 4          ; NUMBER OF WORDS TO MATCH
5487      <1>      ;repe     cmpsw          ; COMPARE THE 8 BYTES AS WORDS
5488      00002AE2 A7      <1>      cmpsd          ; compare first 4 bytes
5489      00002AE3 7501      <1>      jne          ;
5490      00002AE5 A7      <1>      cmpsd          ; compare last 4 bytes
5491      <1>      S17:
5492      00002AE6 5F      <1>      pop      edi          ; RECOVER THE POINTERS
5493      00002AE7 5E      <1>      pop      esi
5494      <1>      ;jz       short S18          ; IF ZERO FLAG SET, THEN MATCH OCCURRED
5495      00002AE8 7406      <1>      je       short S18
5496      <1>      ;
5497      00002AEA 83C708      <1>      add      edi, 8          ; NO MATCH, MOVE ON TO NEXT
5498      <1>      ;dec      ax          ; NEXT CODE POINT
5499      <1>      ; 03/08/2022          ; LOOP CONTROL
5500      00002AED 48      <1>      dec      eax
5501      00002AEE 75F0      <1>      jnz      short S16          ; DO ALL OF THEM
5502      <1>
5503      <1>      ;----- CHARACTER IS FOUND ( AL=0 IF NOT FOUND )
5504      <1>      S18:
5505      00002AF0 83C408      <1>      add      esp, 8          ; READJUST THE STACK, THROW AWAY SAVE

```

```

5506 00002AF3 C3      <1>      retn                ; ALL DONE
5507                  <1>
5508                  <1> ; 12/04/2021
5509                  <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
5510                  <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5511                  <1> ;-----
5512                  <1> ; EXPAND BYTE
5513                  <1> ; THIS ROUTINE TAKES THE BYTE IN AL AND DOUBLES ALL
5514                  <1> ; OF THE BITS, TURNING THE 8 BITS INTO 16 BITS.
5515                  <1> ; THE RESULT IS LEFT IN AX
5516                  <1> ;-----
5517                  <1> S21:
5518                  <1> ; 03/08/2022
5519                  <1> ;push cx                ; SAVE REGISTER
5520                  <1> ; 12/04/2021
5521 00002AF4 51      <1> push ecx
5522                  <1> ;mov cx, 8                ; SHIFT COUNT REGISTER FOR ONE BYTE
5523                  <1> ;mov cl, 8
5524 00002AF5 B408    <1> mov ah, 8
5525                  <1> S22:
5526 00002AF7 D0C8    <1> ror al, 1                ; SHIFT BITS, LOW BIT INTO CARRY FLAG
5527                  <1> ;rcr bp, 1                ; MOVE CARRY FLAG (LOW BIT INTO RESULTS
5528                  <1> ;sar bp, 1                ; SIGN EXTEND HIGH BIT (DOUBLE IT)
5529                  <1> ; 03/08/2022
5530 00002AF9 66D1D9  <1> rcr cx, 1
5531 00002AFC 66D1F9  <1> sar cx, 1
5532                  <1>
5533                  <1> ;loop S22                ; REPEAT FOR ALL 8 BITS
5534                  <1> ;dec cl
5535                  <1> ;jnz short S22
5536                  <1> ;xchg ax, bp                ; MOVE RESULTS TO PARAMETER REGISTER
5537                  <1> ; 03/08/5022
5538 00002AFF FECC    <1> dec ah
5539 00002B01 75F4    <1> jnz short S22
5540 00002B03 6689C8  <1> mov ax, cx
5541                  <1> ;pop cx                ; RECOVER REGISTER
5542                  <1> ; 12/04/2021
5543 00002B06 59      <1> pop ecx
5544 00002B07 C3      <1> retn                ; ALL DONE
5545                  <1>
5546                  <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
5547                  <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5548                  <1> ;-----
5549                  <1> ; MED_READ_BYTE
5550                  <1> ; THIS ROUTINE WILL TAKE 2 BYTES FROM THE REGEN BUFFER,
5551                  <1> ; COMPARE AGAINST THE CURRENT FOREGROUND COLOR, AND PLACE
5552                  <1> ; THE CORRESPONDING ON/OFF BIT PATTERN INTO THE CURRENT
5553                  <1> ; POSITION IN THE SAVE AREA
5554                  <1> ; ENTRY --
5555                  <1> ; SI,DS = POINTER TO REGEN AREA OF INTEREST
5556                  <1> ; BX = EXPANDED FOREGROUND COLOR
5557                  <1> ; BP = POINTER TO SAVE AREA
5558                  <1> ; EXIT --
5559                  <1> ; SI AND BP ARE INCREMENTED
5560                  <1> ;-----
5561                  <1> S23:
5562 00002B08 66AD    <1> lodsw                ; GET FIRST BYTE AND SECOND BYTES
5563 00002B0A 86E0    <1> xchg al, ah          ; SWAP FOR COMPARE
5564                  <1> ;mov cx, 0C000h        ; 2 BIT MASK TO TEST THE ENTRIES
5565                  <1> ; 02/08/2022
5566 00002B0C 29C9    <1> sub ecx, ecx
5567 00002B0E B5C0    <1> mov ch, 0C0h
5568                  <1> ; ecx = 0C000h
5569                  <1> ;mov dl, 0                ; RESULT REGISTER
5570                  <1> ; 03/08/2022
5571 00002B10 28D2    <1> sub dl, dl
5572                  <1> S24:
5573                  <1> ;test ax, cx                ; IS THIS SECTION BACKGROUND?
5574                  <1> ; 03/08/2022
5575 00002B12 85C8    <1> test eax, ecx
5576 00002B14 7401    <1> jz short S25          ; IF ZERO, IT IS BACKGROUND (CARRY=0)
5577 00002B16 F9      <1> stc                ; WASN'T, SO SET CARRY
5578                  <1> S25:
5579 00002B17 D0D2    <1> rcl dl, 1            ; MOVE THAT BIT INTO THE RESULT
5580                  <1> ;shr cx, 2                ; MOVE THE MASK TO THE RIGHT BY 2 BITS
5581                  <1> ; 02/08/2022
5582 00002B19 C1E902  <1> shr ecx, 2
5583 00002B1C 73F4    <1> jnc short S24        ; DO IT AGAIN IF MASK DIDN'T FALL OUT
5584 00002B1E 885500  <1> mov [ebp], dl        ; STORE RESULT IN SAVE AREA
5585 00002B21 45      <1> inc ebp             ; ADJUST POINTER
5586 00002B22 C3      <1> retn                ; ALL DONE
5587                  <1>
5588                  <1> ; 02/08/2022 - TRDOS 386 Kernel v2.0.5
5589                  <1> ; 30/06/2016 - TRDOS 386 (TRDOS v2.0)
5590                  <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5591                  <1> ;-----
5592                  <1> ; V4_POSITION
5593                  <1> ; THIS ROUTINE TAKES THE CURSOR POSITION CONTAINED IN
5594                  <1> ; THE MEMORY LOCATION, AND CONVERTS IT INTO AN OFFSET
5595                  <1> ; INTO THE REGEN BUFFER, ASSUMING ONE BYTE/CHAR.
5596                  <1> ; FOR MEDIUM RESOLUTION GRAPHICS, THE NUMBER MUST
5597                  <1> ; BE DOUBLED.
5598                  <1> ; ENTRY -- NO REGISTERS, MEMORY LOCATION @CURSOR_POSN IS USED
5599                  <1> ; EXIT--
5600                  <1> ; AX CONTAINS OFFSET INTO REGEN BUFFER
5601                  <1> ;-----
5602                  <1> S26:
5603 00002B23 0FB705[BE760100] <1> movzx eax, word [CURSOR_POSN] ; GET CURRENT CURSOR
5604                  <1> GRAPH_POSN:
5605 00002B2A 53      <1> push ebx            ; SAVE REGISTER
5606 00002B2B 0FB6D8  <1> movzx ebx, al        ; SAVE A COPY OF CURRENT CURSOR
5607 00002B2E A0[20680000] <1> mov al, [CRT_COLS]   ; GET BYTES PER COLUMN
5608 00002B33 F6E4    <1> mul ah              ; MULTIPLY BY ROWS
5609                  <1> ;shl ax, 2                ; MULTIPLY * 4 SINCE 4 ROWS/BYTE
5610                  <1> ; 02/08/2022
5611 00002B35 C1E002  <1> shl eax, 2
5612 00002B38 01D8    <1> add eax, ebx         ; DETERMINE OFFSET
5613 00002B3A 5B      <1> pop ebx             ; RECOVER POINTER
5614 00002B3B C3      <1> retn                ; ALL DONE
5615                  <1>
5616                  <1> ; 03/08/2022 - TRDOS 386 Kernel v2.0.5
5617                  <1> ; 09/07/2016
5618                  <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
5619                  <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5620                  <1> ;-----
5621                  <1> ; SET_COLOR
5622                  <1> ; THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE OVERSCAN COLOR,
5623                  <1> ; AND THE FOREGROUND COLOR SET FOR MEDIUM RESOLUTION GRAPHICS
5624                  <1> ; INPUT
5625                  <1> ; (BH) HAS COLOR ID
5626                  <1> ; IF BH=0, THE BACKGROUND COLOR VALUE IS SET
5627                  <1> ; FROM THE LOW BITS OF BL (0-31)
5628                  <1> ; IF BH=1, THE PALETTE SELECTION IS MADE
5629                  <1> ; BASED ON THE LOW BIT OF BL:

```

```

5630 <1> ; 0 = GREEN, RED, YELLOW FOR COLORS 1,2,3
5631 <1> ; 1 = BLUE, CYAN, MAGENTA FOR COLORS 1,2,3
5632 <1> ; (BL) HAS THE COLOR VALUE TO BE USED
5633 <1> ; OUTPUT
5634 <1> ; THE COLOR SELECTION IS UPDATED
5635 <1> ;-----
5636 <1> SET_COLOR:
5637 00002B3C 803D[1E680000]07 <1> cmp byte [CRT_MODE], 7 ; 09/07/2016
5638 <1> ;ja VIDEO_RETURN ; nothing to do for VGA modes
5639 <1> ; 03/08/2022
5640 00002B43 7605 <1> jna short M21
5641 00002B45 E92EF0FFFF <1> jmp VIDEO_RETURN
5642 <1> M21:
5643 <1> ;mov dx, [ADDR_6845] ; I/O PORT FOR PALETTE
5644 <1> ;mov dx, 3D4h
5645 <1> ;add dx, 5 ; OVERSCAN PORT
5646 00002B4A 66BAD903 <1> mov dx, 3D9h
5647 00002B4E A0[21680000] <1> mov al, [CRT_PALETTE] ; GET THE CURRENT PALETTE VALUE
5648 00002B53 08FF <1> or bh, bh ; IS THIS COLOR 0?
5649 00002B55 7512 <1> jnz short M20 ; OUTPUT COLOR 1
5650 <1>
5651 <1> ;----- HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
5652 <1>
5653 00002B57 24E0 <1> and al, 0E0h ; TURN OFF LOW 5 BITS OF CURRENT
5654 00002B59 80E31F <1> and bl, 01Fh ; TURN OFF HIGH 3 BITS OF INPUT VALUE
5655 00002B5C 08D8 <1> or al, bl ; PUT VALUE INTO REGISTER
5656 <1> M19:
5657 00002B5E EE <1> out dx, al ; OUTPUT THE PALETTE
5658 00002B5F A2[21680000] <1> mov [CRT_PALETTE], al ; OUTPUT COLOR SELECTION TO 3D9 PORT
5659 00002B64 E90FF0FFFF <1> jmp VIDEO_RETURN ; SAVE THE COLOR VALUE
5660 <1>
5661 <1> ;----- HANDLE COLOR 1 BY SELECTING THE PALETTE TO BE USED
5662 <1>
5663 <1> M20:
5664 00002B69 24DF <1> and al, 0DFH ; TURN OFF PALETTE SELECT BIT
5665 00002B6B D0EB <1> shr bl, 1 ; TEST THE LOW ORDER BIT OF BL
5666 00002B6D 73EF <1> jnc short M19 ; ALREADY DONE
5667 00002B6F 0C20 <1> or al, 20H ; TURN ON PALETTE SELECT BIT
5668 00002B71 EBEB <1> jmp short M19 ; GO DO IT
5669 <1>
5670 <1> ; 03/08/2022 - TRDOS 386 Kernel v2.0.5
5671 <1> ; 09/07/2016
5672 <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
5673 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5674 <1> ;-----
5675 <1> ; READ DOT -- WRITE DOT
5676 <1> ; THESE ROUTINES WILL WRITE A DOT, OR READ THE
5677 <1> ; DOT AT THE INDICATED LOCATION
5678 <1> ; ENTRY --
5679 <1> ; DX = ROW (0-199) (THE ACTUAL VALUE DEPENDS ON THE MODE)
5680 <1> ; CX = COLUMN ( 0-639) ( THE VALUES ARE NOT RANGE CHECKED )
5681 <1> ; AL = DOT VALUE TO WRITE (1,2 OR 4 BITS DEPENDING ON MODE,
5682 <1> ; REQUIRED FOR WRITE DOT ONLY, RIGHT JUSTIFIED)
5683 <1> ; BIT 7 OF AL = 1 INDICATES XOR THE VALUE INTO THE LOCATION
5684 <1> ; DS = DATA SEGMENT
5685 <1> ; ES = REGEN SEGMENT
5686 <1> ;
5687 <1> ; EXIT
5688 <1> ; AL = DOT VALUE READ, RIGHT JUSTIFIED, READ ONLY
5689 <1> ;-----
5690 <1>
5691 <1> READ_DOT:
5692 <1> ; 09/07/2016
5693 00002B73 8A25[1E680000] <1> mov ah, [CRT_MODE]
5694 00002B79 80FC07 <1> cmp ah, 7 ; 6!?
5695 00002B7C 760A <1> jna short read_dot_cga
5696 <1>
5697 00002B7E E8B3030000 <1> call vga_read_pixel
5698 <1> ; al = pixel value
5699 <1> read_dot_retn: ; 03/08/2022
5700 00002B83 E9F5EFFFFF <1> jmp _video_return
5701 <1>
5702 <1> read_dot_cga:
5703 <1> ;je VIDEO_RETURN ; 7
5704 00002B88 80FC04 <1> cmp ah, 4 ; graphics ?
5705 <1> ;jb VIDEO_RETURN ; no, text mode, nothing to do
5706 <1> ; 03/08/2022
5707 00002B8B 72F6 <1> jb short read_dot_retn
5708 <1>
5709 00002B8D E84F000000 <1> call R3 ; DETERMINE BYTE POSITION OF DOT
5710 00002B92 8A06 <1> mov al, [esi] ; GET THE BYTE
5711 00002B94 20E0 <1> and al, ah ; MASK OFF THE OTHER BITS IN THE BYTE
5712 00002B96 D2E0 <1> shl al, cl ; LEFT JUSTIFY THE VALUE
5713 00002B98 88F1 <1> mov cl, dh ; GET NUMBER OF BITS IN RESULT
5714 00002B9A D2C0 <1> rol al, cl ; RIGHT JUSTIFY THE RESULT
5715 <1> ;jmp VIDEO_RETURN ; RETURN FROM VIDEO I/O
5716 00002B9C 0FB6C0 <1> movzx eax, al
5717 00002B9F E9D9EFFFFF <1> jmp _video_return
5718 <1>
5719 <1> ; 03/08/2022
5720 <1> ; 02/08/2022 - TRDOS 386 Kernel v2.0.5
5721 <1> ; 12/04/2021
5722 <1> ; 09/07/2016
5723 <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
5724 <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5725 <1>
5726 <1> WRITE_DOT:
5727 <1> ; 09/07/2016
5728 00002BA4 8A25[1E680000] <1> mov ah, [CRT_MODE]
5729 00002BAA 80FC07 <1> cmp ah, 7 ; 6!?
5730 00002BAD 760A <1> jna short write_dot_cga
5731 <1>
5732 00002BAF E8F2020000 <1> call vga_write_pixel
5733 <1> write_dot_retn: ; 03/08/2022
5734 00002BB4 E9BFEFFFFF <1> jmp VIDEO_RETURN
5735 <1>
5736 <1> write_dot_cga:
5737 <1> ;je VIDEO_RETURN ; 7
5738 00002BB9 80FC04 <1> cmp ah, 4 ; graphics ?
5739 <1> ;jb VIDEO_RETURN ; no, text mode, nothing to do
5740 <1> ; 03/08/2022
5741 00002BBC 72F6 <1> jb short write_dot_retn
5742 <1>
5743 <1> ;push ax ; SAVE DOT VALUE
5744 <1> ;push ax ; TWICE
5745 <1> ; 12/04/2021
5746 00002BBE 50 <1> push eax
5747 00002BBF E81D000000 <1> call R3 ; DETERMINE BYTE POSITION OF THE DOT
5748 00002BC4 D2E8 <1> shr al, cl ; SHIFT TO SET UP THE BITS FOR OUTPUT
5749 00002BC6 20E0 <1> and al, ah ; STRIP OFF THE OTHER BITS
5750 00002BC8 8A0E <1> mov cl, [esi] ; GET THE CURRENT BYTE
5751 <1> ;pop bx ; RECOVER XOR FLAG
5752 <1> ; 12/04/2021
5753 00002BCA 5B <1> pop ebx

```



```

5754 00002BCB F6C380      <1>      test    bl, 80h          ; IS IT ON
5755 00002BCE 750D      <1>      jnz     short R2        ; YES, XOR THE DOT
5756 00002BD0 F6D4      <1>      not     ah             ; SET MASK TO REMOVE THE INDICATED BITS
5757 00002BD2 20E1      <1>      and     cl, ah
5758 00002BD4 08C8      <1>      or      al, cl          ; OR IN THE NEW VALUE OF THOSE BITS
5759                                <1>      ; FINISH_DOT
5760 00002BD6 8806      <1> R1:    mov     [esi], al      ; RESTORE THE BYTE IN MEMORY
5761                                <1>      ;;;pop    AX
5762 00002BD8 E99BEFFFF    <1>      jmp     VIDEO_RETURN      ; RETURN FROM VIDEO I/O
5763                                <1> R2:    ; XOR_DOT
5764 00002BDD 30C8      <1>      xor     al, cl          ; EXCLUSIVE OR THE DOTS
5765 00002BDF EBF5      <1>      jmp     short R1        ; FINISH UP THE WRITING
5766                                <1>
5767                                <1> ; 02/08/2022 - TRDOS 386 Kernel v2.0.5
5768                                <1> ; 01/07/2016 - TRDOS 386 (TRDOS v2.0)
5769                                <1> ; VIDEO1.ASM - 24/03/1985 (IBM PC-AT BIOS source code)
5770                                <1>
5771                                <1> ;-----
5772                                <1> ; THIS SUBROUTINE DETERMINES THE REGEN BYTE LOCATION OF THE
5773                                <1> ; INDICATED ROW COLUMN VALUE IN GRAPHICS MODE.
5774                                <1> ; ENTRY --
5775                                <1> ; DX = ROW VALUE (0-199)
5776                                <1> ; CX = COLUMN VALUE (0-639)
5777                                <1> ; EXIT --
5778                                <1> ; SI = OFFSET INTO REGEN BUFFER FOR BYTE OF INTEREST
5779                                <1> ; AH = MASK TO STRIP OFF THE BITS OF INTEREST
5780                                <1> ; CL = BITS TO SHIFT TO RIGHT JUSTIFY THE MASK IN AH
5781                                <1> ; DH = # BITS IN RESULT
5782                                <1> ; BX = MODIFIED
5783                                <1> ;-----
5784                                <1> R3:
5785                                <1>
5786                                <1> ;----- DETERMINE 1ST BYTE IN INDICATED ROW BY MULTIPLYING ROW VALUE BY 40
5787                                <1> ;----- ( LOW BIT OF ROW DETERMINES EVEN/ODD, 80 BYTES/ROW )
5788                                <1>
5789 00002BE1 0FB7F0      <1>      movzx   esi, ax          ; WILL SAVE AL AND AH DURING OPERATION
5790 00002BE4 B028      <1>      mov     al, 40
5791 00002BE6 F6E2      <1>      mul     dl             ; AX= ADDRESS OF START OF INDICATED ROW
5792 00002BE8 A808      <1>      test    al, 08h        ; TEST FOR EVEN/ODD ROW CALCULATED
5793 00002BEA 7404      <1>      jz      short R4        ; JUMP IF EVEN ROW
5794 00002BEC 6605D81F    <1>      add     ax, 2000h-40      ; OFFSET TO LOCATION OF ODD ROWS ADJUST
5795                                <1> R4:    ; EVEN_ROW
5796                                <1>      ;xchg    si, ax          ; MOVE POINTER TO (SI) AND RECOVER (AX)
5797                                <1>      ; 02/08/2022
5798 00002BF0 96          <1>      xchg     esi, eax
5799 00002BF1 81C600800B00 <1>      add     esi, 0B8000h
5800                                <1>      ;mov     dx, cx          ; COLUMN VALUE TO DX
5801 00002BF7 0FB7D1      <1>      movzx   edx, cx
5802                                <1>
5803                                <1> ;----- DETERMINE GRAPHICS MODE CURRENTLY IN EFFECT
5804                                <1>
5805                                <1> ; SET UP THE REGISTERS ACCORDING TO THE MODE
5806                                <1> ; CH = MASK FOR LOW OF COLUMN ADDRESS ( 7/3 FOR HIGH/MED RES )
5807                                <1> ; CL = # OF ADDRESS BITS IN COLUMN VALUE ( 3/2 FOR H/M )
5808                                <1> ; BL = MASK TO SELECT BITS FROM POINTED BYTE ( 80H/C0H FOR H/M )
5809                                <1> ; BH = NUMBER OF VALID BITS IN POINTED BYTE ( 1/2 FOR H/M )
5810                                <1>
5811 00002BFA 66B8C002      <1>      mov     bx, 2C0h
5812 00002BFE 66B90203      <1>      mov     cx, 302h        ; SET PARMS FOR MED RES
5813 00002C02 803D[1E680000]06 <1>      cmp     byte [CRT_MODE], 6
5814 00002C09 7208      <1>      jc      short R5        ; HANDLE IF MED RES
5815 00002C0B 66B88001      <1>      mov     bx, 180h
5816 00002C0F 66B90307      <1>      mov     cx, 703h        ; SET PARMS FOR HIGH RES
5817                                <1>
5818                                <1> ;----- DETERMINE BIT OFFSET IN BYTE FROM COLUMN MASK
5819                                <1> R5:
5820 00002C13 20D5      <1>      and     ch, dl          ; ADDRESS OF PEL WITHIN BYTE TO CH
5821                                <1>
5822                                <1> ;----- DETERMINE BYTE OFFSET FOR THIS LOCATION IN COLUMN
5823                                <1>
5824                                <1>      ;shr     dx, cl          ; SHIFT BY CORRECT AMOUNT
5825                                <1>      ;add     si, dx          ; INCREMENT THE POINTER
5826                                <1>      ; 02/08/2022
5827 00002C15 D3EA      <1>      shr     edx, cl
5828 00002C17 01D6      <1>      add     esi, edx
5829 00002C19 88FE      <1>      mov     dh, bh          ; GET THE # OF BITS IN RESULT TO DH
5830                                <1>
5831                                <1> ;----- MULTIPLY BH (VALID BITS IN BYTE) BY CH (BIT OFFSET)
5832                                <1>
5833 00002C1B 28C9      <1>      sub     cl, cl          ; ZERO INTO STORAGE LOCATION
5834                                <1> R6:
5835 00002C1D D0C8      <1>      ror     al, 1           ; LEFT JUSTIFY VALUE IN AL (FOR WRITE)
5836 00002C1F 00E9      <1>      add     cl, ch          ; ADD IN THE BIT OFFSET VALUE
5837 00002C21 FECF      <1>      dec     bh             ; LOOP CONTROL
5838 00002C23 75F8      <1>      jnz     short R6        ; ON EXIT, CL HAS COUNT TO RESTORE BITS
5839 00002C25 88DC      <1>      mov     ah, bl          ; GET MASK TO AH
5840 00002C27 D2EC      <1>      shr     ah, cl          ; MOVE THE MASK TO CORRECT LOCATION
5841 00002C29 C3          <1>      retn          ; RETURN WITH EVERYTHING SET UP
5842                                <1>
5843                                <1> load_dac_palette:
5844                                <1> ; 02/08/2022 (TRDOS 386 Kernel v2.0.5)
5845                                <1> ; 29/07/2016
5846                                <1> ; 23/07/2016
5847                                <1> ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
5848                                <1> ; (set_mode_vga)
5849                                <1> ; derived from 'Plex86/Bochs VGABios' source code
5850                                <1> ; vgabios-0.7a (2011)
5851                                <1> ; by the LGPL VGABios developers Team (2001-2008)
5852                                <1> ; 'vgabios.c', 'load_dac_palette'
5853                                <1> ;
5854                                <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code
5855                                <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
5856                                <1> ;
5857                                <1> ; INPUT -> AH = DAC selection number (3, 2 or 1)
5858                                <1> ; OUTPUT -> ECX = 0, AX = 0
5859                                <1> ; (Modified registers: EAX, ECX, EDX, ESI)
5860                                <1> ;
5861 00002C2A 66BAC803      <1>      mov     dx, 3C8h        ; VGAREG_DAC_WRITE_ADDRESS
5862 00002C2E 28C0      <1>      sub     al, al          ; 0
5863 00002C30 EE          <1>      out     dx, al          ; 0 ; color index, always 0 at the beginning
5864                                <1>      ;inc     dx          ; 3C9h ; VGAREG_DAC_DATA
5865                                <1>      ; 02/08/2022
5866 00002C31 FEC2      <1>      inc     dl          ; dx = 3C9h
5867                                <1>      ;mov     ecx, 256        ; always 256*3 values
5868                                <1>      ; 02/08/2022
5869 00002C33 31C9      <1>      xor     ecx, ecx
5870 00002C35 FEC5      <1>      inc     ch
5871                                <1>      ; ecx = 256
5872                                <1>
5873                                <1>      ;push    esi
5874 00002C37 88E0      <1>      mov     al, ah
5875 00002C39 B43F      <1>      mov     ah, 3Fh        ; 3Fh except DAC selection number 3
5876 00002C3B 3C02      <1>      cmp     al, 2
5877 00002C3D 7414      <1>      je      short l_dac_p_2

```

```

5878 00002C3F 7719      <1>      ja      short 1_dac_p_3
5879 00002C41 20C0      <1>      and     al, al
5880 00002C43 7507      <1>      jnz     short 1_dac_p_1
5881                                <1> 1_dac_p_0:
5882 00002C45 BE[F0470100] <1>      mov     esi, palette0
5883 00002C4A EB15      <1>      jmp     short 1_dac_p_4
5884                                <1> 1_dac_p_1:
5885 00002C4C BE[B0480100] <1>      mov     esi, palette1
5886 00002C51 EB0E      <1>      jmp     short 1_dac_p_4
5887                                <1> 1_dac_p_2:
5888 00002C53 BE[70490100] <1>      mov     esi, palette2
5889 00002C58 EB07      <1>      jmp     short 1_dac_p_4
5890                                <1> 1_dac_p_3:
5891 00002C5A B4FF      <1>      mov     ah, 0FFh ; dac registers
5892 00002C5C BE[304A0100] <1>      mov     esi, palette3
5893                                <1> 1_dac_p_4:
5894 00002C61 AC          <1>      lodsb
5895 00002C62 EE          <1>      out     dx, al ; Red
5896 00002C63 AC          <1>      lodsb
5897 00002C64 EE          <1>      out     dx, al ; Green
5898 00002C65 AC          <1>      lodsb
5899 00002C66 EE          <1>      out     dx, al ; Blue
5900 00002C67 20E4      <1>      and     ah, ah
5901 00002C69 7405      <1>      jz      short 1_dac_p_5
5902 00002C6B FECC      <1>      dec     ah
5903 00002C6D E2F2      <1>      loop    1_dac_p_4
5904                                <1>      ;pop     esi
5905 00002C6F C3          <1>      retn
5906                                <1> 1_dac_p_5:
5907                                <1>      ; 29/07/2016
5908 00002C70 FEC9      <1>      dec     cl
5909 00002C72 7407      <1>      jz      short 1_dac_p_7
5910                                <1>      ;
5911 00002C74 28C0      <1>      sub     al, al ; 0
5912                                <1> 1_dac_p_6:
5913 00002C76 EE          <1>      out     dx, al ; outb(VGAREG_DAC_DATA,0);
5914 00002C77 EE          <1>      out     dx, al
5915 00002C78 EE          <1>      out     dx, al
5916 00002C79 E2FB      <1>      loop    1_dac_p_6
5917                                <1> 1_dac_p_7:
5918                                <1>      ;pop     esi
5919 00002C7B C3          <1>      retn
5920                                <1>
5921                                <1> gray_scale_summing:
5922                                <1>      ; 03/08/2022 (TRDOS 386 v2.0.5)
5923                                <1>      ; 12/04/2021
5924                                <1>      ; 03/07/2016 (TRDOS 386 = TRDOS v2.0)
5925                                <1>      ; (set_mode_vga)
5926                                <1>      ; derived from 'Plex86/Bochs VGABios' source code
5927                                <1>      ; vgabios-0.7a (2011)
5928                                <1>      ; by the LGPL VGABios developers Team (2001-2008),
5929                                <1>      ; 'vgabios.c', 'biosfn_perform_gray_scale_summing',
5930                                <1>      ;
5931                                <1>      ; Oracle VirtualBox 5.0.24 VGABios Source Code
5932                                <1>      ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
5933                                <1>      ;
5934                                <1>
5935                                <1>      ; INPUT -> EBX = Start address (color index <= 255)
5936                                <1>      ;      ECX = Count (<= 256)
5937                                <1>      ; OUTPUT -> (E)CX = 0
5938                                <1>      ; (Modified registers: EAX, ECX, EDX, EBX)
5939                                <1>
5940 00002C7C 66BADA03    <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
5941 00002C80 EC          <1>      in      al, dx
5942 00002C81 30C0      <1>      xor     al, al ; 0
5943                                <1>      ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
5944                                <1>      ; 03/08/2022
5945 00002C83 B2C0      <1>      mov     dl, 0C0h
5946 00002C85 EE          <1>      out     dx, al ; clear bit 5
5947                                <1>      ; (while loading palette registers)
5948                                <1>      ; set read address and switch to read mode
5949                                <1> g_s_s_1:
5950                                <1>      ;mov     dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
5951                                <1>      ; 03/08/2022
5952 00002C86 B2C7      <1>      mov     dl, 0C7h
5953 00002C88 88D8      <1>      mov     al, bl
5954 00002C8A EE          <1>      out     dx, al
5955                                <1>      ; get 6-bit wide RGB data values
5956                                <1>      ; intensity = (0.3*Red)+(0.59*Green)+(0.11*Blue)
5957                                <1>      ; i = ( ( 77*r + 151*g + 28*b ) + 0x80 ) >> 8;
5958                                <1>      ;mov     dx, 3C9h ; VGAREG_DAC_DATA
5959                                <1>      ; 03/08/2022
5960 00002C8B B2C9      <1>      mov     dl, 0C9h
5961 00002C8D EC          <1>      in      al, dx ; red
5962 00002C8E B44D      <1>      mov     ah, 77 ; 0.3* Red
5963 00002C90 F6E4      <1>      mul     ah
5964                                <1>      ;push     ax
5965                                <1>      ; 12/04/2021
5966 00002C92 50          <1>      push    eax
5967 00002C93 EC          <1>      in      al, dx ; green
5968 00002C94 B497      <1>      mov     ah, 151 ; 0.59 * Green
5969 00002C96 F6E4      <1>      mul     ah
5970                                <1>      ;push     ax
5971                                <1>      ; 12/04/2021
5972 00002C98 50          <1>      push    eax
5973 00002C99 EC          <1>      in      al, dx ; blue
5974 00002C9A B41C      <1>      mov     ah, 28 ; 0.11 * Blue
5975 00002C9C F6E4      <1>      mul     ah
5976                                <1>      ;pop     dx
5977                                <1>      ; 12/04/2021
5978 00002C9E 5A          <1>      pop     edx
5979 00002C9F 6601D0      <1>      add     ax, dx
5980                                <1>      ;pop     dx
5981                                <1>      ; 12/04/2021
5982 00002CA2 5A          <1>      pop     edx
5983 00002CA3 6601D0      <1>      add     ax, dx
5984 00002CA6 66058000    <1>      add     ax, 80h
5985 00002CAA B03F      <1>      mov     al, 3Fh
5986 00002CAC 38C4      <1>      cmp     ah, al ; if(i>0x3f)i=0x3f
5987 00002CAE 7602      <1>      jna     short g_s_s_2
5988 00002CB0 88C4      <1>      mov     ah, al
5989                                <1> g_s_s_2:
5990                                <1>      ;mov     dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
5991                                <1>      ; 03/08/2022
5992 00002CB2 B2C8      <1>      mov     dl, 0C8h
5993 00002CB4 88D8      <1>      mov     al, bl ; color index
5994 00002CB6 EE          <1>      out     dx, al
5995 00002CB7 88E0      <1>      mov     al, ah ; intensity
5996                                <1>      ;inc     dx ; 3C9h ; VGAREG_DAC_DATA
5997                                <1>      ; 03/08/2022
5998 00002CB9 FEC2      <1>      inc     dl
5999 00002CBB EE          <1>      out     dx, al ; R (R=G=B)
6000 00002CBC 88E0      <1>      mov     al, ah ; intensity
6001 00002CBE EE          <1>      out     dx, al ; G (R=G=B)

```

```

6002 00002CBF 88E0      <1>      mov     al, ah ; intensity
6003 00002CC1 EE        <1>      out     dx, al ; B (R=G=B)
6004                      <1>      ;dec     cx
6005                      <1>      ; 03/08/2022
6006 00002CC2 49        <1>      dec     ecx
6007 00002CC3 7404      <1>      jz      short g_s_s_3
6008 00002CC5 FEC3      <1>      inc     bl ; next color index value
6009 00002CC7 EBBD      <1>      jmp     short g_s_s_1
6010                      <1>      g_s_s_3:
6011                      <1>      ;mov     dx, 3DAh ; VGAREG_ACTL_RESET
6012                      <1>      ; 03/08/2022
6013 00002CC9 B2DA      <1>      mov     dl, 0DAh
6014 00002CCB EC         <1>      in      al, dx
6015 00002CCC B020      <1>      mov     al, 20h
6016                      <1>      ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
6017                      <1>      ; 03/08/2022
6018 00002CCE B2C0      <1>      mov     dl, 0C0h
6019 00002CD0 EE        <1>      out     dx, al ; 20h -> set bit 5
6020                      <1>      ; (after loading palette regs)
6021 00002CD1 C3         <1>      retn
6022                      <1>
6023                      <1>      vga_write_char_attr:
6024                      <1>      vga_write_char_only:
6025                      <1>      ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
6026                      <1>      ;
6027                      <1>      ; derived from 'Plex86/Bochs VGABios' source code
6028                      <1>      ; vgabios-0.7a (2011)
6029                      <1>      ; by the LGPL VGABios developers Team (2001-2008)
6030                      <1>      ; 'vgabios.c', 'biosfn_write_char_attr'
6031                      <1>      ; 'biosfn_write_char_only'
6032                      <1>
6033                      <1>      ; INPUT ->
6034                      <1>      ; [CRT_MODE] = current video mode (>7)
6035                      <1>      ; CX = Count of characters to write
6036                      <1>      ; AL = Character to write
6037                      <1>      ; BL = Color of character
6038                      <1>      ; OUTPUT ->
6039                      <1>      ; Regen buffer updated
6040                      <1>
6041 00002CD2 8A25[1E680000] <1>      mov     ah, [CRT_MODE]
6042 00002CD8 668B15[BE760100] <1>      mov     dx, [CURSOR_POSN] ; cursor pos for page 0
6043                      <1>
6044 00002CDF BE[3A680000] <1>      mov     esi, vga_modes
6045 00002CE4 89F7        <1>      mov     edi, esi
6046 00002CE6 83C710      <1>      add     edi, vga_mode_count
6047                      <1>      vga_wca_0:
6048 00002CE9 AC         <1>      lodsb
6049 00002CEA 38E0        <1>      cmp     al, ah ; [CRT_MODE]
6050 00002CEC 7405        <1>      je      short vga_wca_2
6051 00002CEE 39FE        <1>      cmp     esi, edi
6052 00002CF0 72F7        <1>      jb      short vga_wca_0
6053                      <1>      vga_wca_1:
6054 00002CF2 C3         <1>      retn ; nothing to do
6055                      <1>      vga_wca_2:
6056 00002CF3 83C64F      <1>      add     esi, vga_memmodel - (vga_modes + 1)
6057                      <1>      ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
6058                      <1>
6059                      <1>      ; biosfn_write_char_attr (car,page,attr,count)
6060                      <1>      ; AL = car, page = 0, BL = attr, CX = count
6061 00002CF6 803E04      <1>      cmp     byte [esi], PLANAR4
6062 00002CF9 741D        <1>      je      short vga_wca_planar
6063 00002CFB 803E03      <1>      cmp     byte [esi], PLANAR1
6064 00002CFE 7418        <1>      je      short vga_wca_planar
6065                      <1>      vga_wca_linear8:
6066                      <1>      ; while((count-->0) && (xcurs<nbcols))
6067                      <1>      ; CX = count
6068 00002D00 6621C9      <1>      and     cx, cx
6069 00002D03 74ED        <1>      jz      short vga_wca_1
6070 00002D05 3A15[20680000] <1>      cmp     dl, [CRT_COLS]
6071 00002D0B 73E5        <1>      jnb     short vga_wca_1
6072                      <1>      ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols);
6073                      <1>      ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
6074                      <1>      ; [CRT_COLS] = nbcols
6075 00002D0D E81E000000 <1>      call    write_gfx_char_lin
6076 00002D12 6649        <1>      dec     cx ; count
6077 00002D14 FEC2        <1>      inc     dl ; xcurs
6078 00002D16 EBE8        <1>      jmp     short vga_wca_linear8
6079                      <1>      vga_wca_planar:
6080                      <1>      ; while((count-->0) && (xcurs<nbcols))
6081                      <1>      ; CX = count
6082 00002D18 6621C9      <1>      and     cx, cx
6083 00002D1B 74D5        <1>      jz      short vga_wca_1
6084 00002D1D 3A15[20680000] <1>      cmp     dl, [CRT_COLS]
6085 00002D23 73CD        <1>      jnb     short vga_wca_1
6086                      <1>      ; write_gfx_char_pl4(car,attr,xcurs,ycurs,nbcols,height);
6087                      <1>      ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
6088                      <1>      ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = height
6089 00002D25 E8A7000000 <1>      call    write_gfx_char_pl4
6090 00002D2A 6649        <1>      dec     cx ; count
6091 00002D2C FEC2        <1>      inc     dl ; xcurs
6092 00002D2E EBE8        <1>      jmp     short vga_wca_planar
6093                      <1>
6094                      <1>      write_gfx_char_lin:
6095                      <1>      ; 02/08/2022 (TRDOS 386 v2.0.5)
6096                      <1>      ; 08/01/2021
6097                      <1>      ; 05/01/2021 (TRDOS 386 v2.0.3)
6098                      <1>      ; 08/08/2016
6099                      <1>      ; 31/07/2016
6100                      <1>      ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
6101                      <1>      ;
6102                      <1>      ; derived from 'Plex86/Bochs VGABios' source code
6103                      <1>      ; vgabios-0.7a (2011)
6104                      <1>      ; by the LGPL VGABios developers Team (2001-2008)
6105                      <1>      ; 'vgabios.c', 'write_gfx_char_lin'
6106                      <1>
6107                      <1>      ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols)
6108                      <1>      ; INPUT ->
6109                      <1>      ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
6110                      <1>      ; [CRT_COLS] = nbcols
6111                      <1>      ; OUTPUT ->
6112                      <1>      ; Regen buffer updated
6113                      <1>
6114 00002D30 51          <1>      push     ecx
6115 00002D31 53          <1>      push     ebx
6116 00002D32 52          <1>      push     edx
6117 00002D33 50          <1>      push     eax
6118                      <1>      ; addr=xcurs*8+ycurs*nbcols*64;
6119                      <1>      ; 08/08/2016
6120 00002D34 0FB6F0      <1>      movzx   esi, al ; car
6121                      <1>      ; 08/01/2021
6122                      <1>      ;movzx   eax, dh ; ycurs
6123                      <1>      ;mov     ah, [CRT_COLS] ; nbcols
6124                      <1>      ;mul     ah
6125 00002D37 A0[20680000] <1>      mov     al, [CRT_COLS]

```

```

6126 00002D3C F6E6      <1>      mul     dh
6127                    <1>      ;shl     ax, 6 ; * 64
6128                    <1>      ;shl     ax, 3 ; 8 * ycurs * [CRT_COLS]
6129                    <1>      ; 02/08/2022
6130 00002D3E C1E003    <1>      shl     eax, 3
6131                    <1>      ;sub     dh, dh
6132                    <1>      ;shl     dx, 3 ; xcurs * 8
6133                    <1>      ;movzx   edi, dx
6134 00002D41 BF0000A0    <1>      mov     edi, 0A0000h
6135 00002D46 30F6      <1>      xor     dh, dh
6136 00002D48 6689D7    <1>      mov     di, dx
6137                    <1>      ;movzx   edi, di
6138 00002D4B 66C1E703   <1>      shl     di, 3 ; xcurs * 8
6139                    <1>      ;xor     dh, dh
6140 00002D4F 8A15[22680000] <1>      mov     di, [CHAR_HEIGHT]
6141 00002D55 66F7E2    <1>      mul     dx
6142                    <1>      ; eax = ycurs*nbcols*8*[CHAR_HEIGHT]
6143                    <1>      ;add     edi, eax ; addr
6144                    <1>      ;add     edi, 0A0000h
6145 00002D58 6601C7    <1>      add     di, ax
6146                    <1>      ;shl     si, 3 ; car * 8
6147 00002D5B 30E4      <1>      xor     ah, ah
6148 00002D5D A0[22680000] <1>      mov     al, [CHAR_HEIGHT]
6149 00002D62 66F7E6    <1>      mul     si
6150 00002D65 6689C6    <1>      mov     si, ax
6151                    <1>      ;; esi = src = car * 8
6152                    <1>      ; esi = src = car * [CHAR_HEIGHT]
6153                    <1>      ; i = 0
6154                    <1>      ;add     esi, vgafont8 ; fdata [src+i]
6155                    <1>      ; 08/08/2016
6156 00002D68 A1[4A830100] <1>      mov     eax, [VGA_INT43H]
6157 00002D6D 09C0      <1>      or      eax, eax ; 0 ?
6158 00002D6F 743E      <1>      jz      short wfxl_7 ; yes, default font
6159                    <1>      ;cmp     eax, vgafont16
6160                    <1>      ;je short wgfxl_0
6161                    <1>      ;cmp     eax, vgafont14
6162                    <1>      ;je short wgfxl_0
6163                    <1>      ;cmp     eax, vgafont8
6164                    <1>      ;je short wgfxl_0
6165                    <1>      ; 05/01/2021 (TRDOS 386 v2.0.3)
6166                    <1>      ; user font
6167                    <1>      ;mov     eax, VGAFONTUSR ; 8x16 or 8x8 or 8x14 font
6168                    <1>      ; (256 characters)
6169                    <1>      wgfxl_0:
6170 00002D71 01C6      <1>      add     esi, eax
6171                    <1>      wgfxl_1:
6172 00002D73 28FF      <1>      sub     bh, bh ; i = 0
6173                    <1>      wgfxl_2:
6174                    <1>      ; for(i=0;i<8;i++)
6175 00002D75 57          <1>      push    edi ; addr
6176 00002D76 0FB605[20680000] <1>      movzx   eax, byte [CRT_COLS] ; nbcols
6177 00002D7D F6E7      <1>      mul     bh ; nbcols*i
6178                    <1>      ;shl     ax, 3 ; i*nbcols*8
6179                    <1>      ; 02/08/2022
6180 00002D7F C1E003    <1>      shl     eax, 3
6181                    <1>      ; dest=addr+i*nbcols*8;
6182 00002D82 01C7      <1>      add     edi, eax ; dest + j ; j = 0
6183 00002D84 B180      <1>      mov     cl, 80h ; mask = 0x80;
6184                    <1>      ; esi = fdata + src + i
6185                    <1>      ; for(j=0;j<8;j++)
6186 00002D86 29D2      <1>      sub     edx, edx ; j = 0
6187                    <1>      wgfxl_3:
6188 00002D88 8A06      <1>      mov     al, [esi] ; al = fdata[src+i]
6189 00002D8A 20C8      <1>      and     al, cl ; if (fdata[src+i] & mask)
6190 00002D8C 7402      <1>      jz      short wgfxl_4 ; data = 0, zf = 1
6191 00002D8E 88D8      <1>      mov     al, bl ; data = attr;
6192                    <1>      wgfxl_4:
6193                    <1>      ; write_byte(0xa000,dest+j,data);
6194 00002D90 AA          <1>      stosb   ; dest + j (+ 0A0000h)
6195                    <1>      ;inc     di ; j++
6196                    <1>      ;cmp     di, 8
6197 00002D91 80FA07    <1>      cmp     di, 7
6198 00002D94 720E      <1>      jb      short wgfxl_5
6199 00002D96 5F          <1>      pop     edi
6200                    <1>      ; 08/08/2016
6201                    <1>      ;cmp     bh, 7
6202                    <1>      ;jnb     short wgfxl_6
6203 00002D97 FEC7      <1>      inc     bh ; i++
6204 00002D99 3A3D[22680000] <1>      cmp     bh, [CHAR_HEIGHT]
6205 00002D9F 7309      <1>      jnb     short wgfxl_6
6206 00002DA1 46          <1>      inc     esi
6207 00002DA2 EBD1      <1>      jmp     short wgfxl_2
6208                    <1>      wgfxl_5:
6209 00002DA4 D0E9      <1>      shr     cl, 1 ; mask >>= 1;
6210 00002DA6 FEC2      <1>      inc     di ; j++
6211 00002DA8 EBDE      <1>      jmp     short wgfxl_3
6212                    <1>      wgfxl_6:
6213 00002DAA 58          <1>      pop     eax
6214 00002DAB 5A          <1>      pop     edx
6215 00002DAC 5B          <1>      pop     ebx
6216 00002DAD 59          <1>      pop     ecx
6217 00002DAE C3          <1>      retn
6218                    <1>      wfxl_7:
6219                    <1>      ; 08/01/2021
6220                    <1>      ; 05/01/2021
6221                    <1>      ; Default font (8x8 or 8x14 or 8x16)
6222 00002DAF A0[22680000] <1>      mov     al, [CHAR_HEIGHT]
6223 00002DB4 3C08      <1>      cmp     al, 8
6224 00002DB6 7507      <1>      jne     short wfxl_8
6225 00002DB8 B8[304D0100] <1>      mov     eax, vgafont8
6226 00002DBD EBB2      <1>      jmp     short wgfxl_0
6227                    <1>      wfxl_8:
6228 00002DBF 3C0E      <1>      cmp     al, 14
6229 00002DC1 7507      <1>      jne     short wfxl_9
6230 00002DC3 B8[30550100] <1>      mov     eax, vgafont14
6231 00002DC8 EBA7      <1>      jmp     short wgfxl_0
6232                    <1>      wfxl_9:
6233 00002DCA B8[30630100] <1>      mov     eax, vgafont16
6234 00002DCF EBA0      <1>      jmp     short wgfxl_0
6235                    <1>
6236                    <1>      write_gfx_char_p14:
6237                    <1>      ; 02/08/2022 (TRDOS 386 kernel v2.0.5)
6238                    <1>      ; 08/08/2016
6239                    <1>      ; 08/07/2016 (TRDOS 386 = TRDOS v2.0)
6240                    <1>      ;
6241                    <1>      ; derived from 'Plex86/Bochs VGABios' source code
6242                    <1>      ; vgabios-0.7a (2011)
6243                    <1>      ; by the LGPL VGABios developers Team (2001-2008)
6244                    <1>      ; 'vgabios.c', 'write_gfx_char_p14'
6245                    <1>
6246                    <1>      ; write_gfx_char_p14(car,attr,xcurs,ycurs,nbcols,height)
6247                    <1>      ; INPUT ->
6248                    <1>      ; AL = car, BL = attr, DL = xcurs, DH = ycurs,
6249                    <1>      ; [CRT_COLS] = nbcols, [CHAR_HEIGHT] = height

```

```

6250      <1>      ; OUTPUT ->
6251      <1>      ; Regen buffer updated
6252      <1>
6253      00002DD1 51      <1>      push    ecx
6254      00002DD2 53      <1>      push    ebx
6255      00002DD3 52      <1>      push    edx
6256      00002DD4 50      <1>      push    eax
6257      <1>      wgfxpl_f0:
6258      <1>      ; switch(cheight)
6259      00002DD5 8A25[22680000] <1>      mov     ah, [CHAR_HEIGHT]
6260      00002DDB 80FC10 <1>      cmp     ah, 16 ; case 16:
6261      00002DDE 7507 <1>      jne     short wgfxpl_f1
6262      <1>      ; fdata = &vgafont16;
6263      00002DE0 BE[30630100] <1>      mov     esi, vgafont16
6264      00002DE5 EB13 <1>      jmp     short wgfxpl_f3
6265      <1>      wgfxpl_f1:
6266      00002DE7 80FC0E <1>      cmp     ah, 14 ; case 14:
6267      00002DEA 7507 <1>      jne     short wgfxpl_f2
6268      00002DEC BE[30550100] <1>      mov     esi, vgafont14
6269      00002DF1 EB07 <1>      jmp     short wgfxpl_f3
6270      <1>      wgfxpl_f2:
6271      <1>      ; default:
6272      <1>      ; fdata = &vgafont8;
6273      00002DF3 BE[304D0100] <1>      mov     esi, vgafont8
6274      00002DF8 B408 <1>      mov     ah, 8
6275      <1>      wgfxpl_f3:
6276      <1>      ; al = car
6277      00002DFA F6E4 <1>      mul     ah ; ah = cheight
6278      00002DFC 25FFFF0000 <1>      and     eax, 0FFFFh ; car * cheight
6279      <1>      ; src = car * cheight;
6280      00002E01 01C6 <1>      add     esi, eax ; esi = fdata[src+i]
6281      <1>      ; addr=xcurs*8+ycurs*nbcols*64;
6282      00002E03 88F0 <1>      mov     al, dh ; ycurs
6283      00002E05 8A25[20680000] <1>      mov     ah, [CRT_COLS] ; nbcols
6284      00002E0B F6E4 <1>      mul     ah
6285      <1>      ; 08/08/2016
6286      <1>      ;shl ax, 6 ; * 64
6287      <1>      ;shl ax, 3 ; * 8
6288      <1>      ; 02/08/2022
6289      00002E0D C1E003 <1>      shl     eax, 3
6290      <1>      ;sub dh, dh ; 0
6291      <1>      ;shl dx, 3 ; xcurs * 8
6292      <1>      ;movzx edi, dx
6293      00002E10 0FB6FA <1>      movzx   edi, dl
6294      <1>      ;shl di, 3 ; xcurs * 8
6295      <1>      ; 02/08/2022
6296      00002E13 C1E703 <1>      shl     edi, 3
6297      00002E16 30F6 <1>      xor     dh, dh
6298      00002E18 8A15[22680000] <1>      mov     dl, [CHAR_HEIGHT]
6299      00002E1E 66F7E2 <1>      mul     dx
6300      <1>      ; eax = ycurs*nbcols*8*[CHAR_HEIGHT]
6301      00002E21 01C7 <1>      add     edi, eax ; addr
6302      00002E23 81C700000A00 <1>      add     edi, 0A0000h
6303      <1>      ;
6304      <1>      ; outw(VGAREG_SEQU_ADDRESS, 0x0f02);
6305      <1>      ; outw(VGAREG_GRDC_ADDRESS, 0x0205);
6306      00002E29 66BAC403 <1>      mov     dx, 3C4h ; VGAREG_SEQU_ADDRESS
6307      00002E2D 66B8020F <1>      mov     ax, 0F02h
6308      00002E31 66EF <1>      out     dx, ax
6309      00002E33 66BACE03 <1>      mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
6310      00002E37 66B80502 <1>      mov     ax, 0205h
6311      00002E3B 66EF <1>      out     dx, ax
6312      <1>      ;
6313      00002E3D 66BACE03 <1>      mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
6314      00002E41 F6C380 <1>      test    bl, 80h ; if(attr&0x80)
6315      00002E44 7406 <1>      jz      short wgfxpl_f4 ; else
6316      <1>      ; outw(VGAREG_GRDC_ADDRESS, 0x1803);
6317      00002E46 66B80318 <1>      mov     ax, 1803h
6318      00002E4A EB04 <1>      jmp     short wgfxpl_f5
6319      <1>      wgfxpl_f4:
6320      <1>      ; outw(VGAREG_GRDC_ADDRESS, 0x0003);
6321      00002E4C 66B80300 <1>      mov     ax, 0003h
6322      <1>      wgfxpl_f5:
6323      00002E50 66EF <1>      out     dx, ax
6324      <1>      ;
6325      00002E52 28FF <1>      sub     bh, bh ; i = 0
6326      <1>      wgfxpl_0:
6327      <1>      ; for(i=0;i<cheight;i++)
6328      00002E54 57 <1>      push    edi ; addr
6329      00002E55 0FB605[20680000] <1>      movzx   eax, byte [CRT_COLS] ; nbcols
6330      00002E5C F6E7 <1>      mul     bh ; nbcols*i
6331      <1>      ; dest=addr+i*nbcols
6332      00002E5E 01C7 <1>      add     edi, eax ; dest
6333      00002E60 B580 <1>      mov     ch, 80h ; mask = 0x80;
6334      <1>      ; for(j=0;j<8;j++)
6335      00002E62 28C9 <1>      sub     cl, cl ; j = 0
6336      <1>      wgfxpl_1:
6337      00002E64 D2ED <1>      shr     ch, cl ; mask=0x80>>j;
6338      <1>      ;
6339      <1>      ; outw(VGAREG_GRDC_ADDRESS, (mask << 8) | 0x08);
6340      <1>      ; read_byte(0xa000,dest);
6341      <1>      ;mov dx, 3CEh ; VGAREG_GRDC_ADDRESS
6342      00002E66 88EC <1>      mov     ah, ch
6343      00002E68 B008 <1>      mov     al, 8
6344      00002E6A 66EF <1>      out     dx, ax
6345      00002E6C 8A07 <1>      mov     al, [edi] ; ? (io delay?)
6346      <1>      ;
6347      00002E6E 28C0 <1>      sub     al, al ; attr = 0
6348      <1>      ; if (fdata[src+i] & mask)
6349      00002E70 842E <1>      test    byte [esi], ch
6350      00002E72 7404 <1>      jz      short wgfxpl_2 ; zf = 1
6351      <1>      ; write_byte(0xa000,dest,attr&0x0f);
6352      00002E74 88D8 <1>      mov     al, bl ; attr;
6353      00002E76 240F <1>      and     al, 0Fh ; attr&0x0f
6354      <1>      wgfxpl_2:
6355      <1>      ; write_byte(0xa000,dest,0x00);
6356      00002E78 8807 <1>      mov     [edi], al ; dest (+ 0A0000h)
6357      00002E7A FEC1 <1>      inc     cl ; j++
6358      00002E7C 80F908 <1>      cmp     cl, 8
6359      00002E7F 72E3 <1>      jb      short wgfxpl_1
6360      00002E81 5F <1>      pop     edi
6361      <1>      ; 08/08/2016
6362      <1>      ;cmp bh, 7
6363      <1>      ;jnb short wgfxpl_3
6364      00002E82 FEC7 <1>      inc     bh ; i++
6365      00002E84 3A3D[22680000] <1>      cmp     bh, [CHAR_HEIGHT]
6366      00002E8A 7303 <1>      jnb     short wgfxpl_3
6367      00002E8C 46 <1>      inc     esi
6368      00002E8D EBC5 <1>      jmp     short wgfxpl_0
6369      <1>      wgfxpl_3:
6370      <1>      ;mov dx, 3CEh ; VGAREG_GRDC_ADDRESS
6371      00002E8F 66B808FF <1>      mov     ax, 0FF08h
6372      00002E93 66EF <1>      out     dx, ax
6373      00002E95 66B80500 <1>      mov     ax, 0005h

```

```

6374 00002E99 66EF      <1>      out      dx, ax
6375 00002E9B 66B80300  <1>      mov      ax, 0003h
6376 00002E9F 66EF      <1>      out      dx, ax
6377                                <1>      ;
6378 00002EA1 58      <1>      pop      eax
6379 00002EA2 5A      <1>      pop      edx
6380 00002EA3 5B      <1>      pop      ebx
6381 00002EA4 59      <1>      pop      ecx
6382 00002EA5 C3      <1>      retn
6383                                <1>
6384                                <1> vga_write_pixel:
6385                                <1>      ; 02/08/2022 (TRDOS 386 Kerbel v2.0.5)
6386                                <1>      ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
6387                                <1>      ;
6388                                <1>      ; derived from 'Plex86/Bochs VGABios' source code
6389                                <1>      ; vgabios-0.7a (2011)
6390                                <1>      ; by the LGPL VGABios developers Team (2001-2008)
6391                                <1>      ; 'vgabios.c', 'biosfn_write_pixel'
6392                                <1>
6393                                <1>      ; INPUT ->
6394                                <1>      ; DX = row (0-239)
6395                                <1>      ; CX = column (0-799)
6396                                <1>      ; AL = pixel value
6397                                <1>      ; (AH = [CRT_MODE])
6398                                <1>      ; OUTPUT ->
6399                                <1>      ; none
6400                                <1>
6401 00002EA6 88C3      <1>      mov      bl, al ; pixel value
6402                                <1>      ;mov      ah, [CRT_MODE]
6403 00002EA8 BE[3A680000] <1>      mov      esi, vga_modes
6404 00002EAD 89F7      <1>      mov      edi, esi
6405 00002EAF 83C710      <1>      add      edi, vga_mode_count
6406                                <1> vga_wp_0:
6407 00002EB2 AC      <1>      lodsb
6408 00002EB3 38E0      <1>      cmp      al, ah ; [CRT_MODE]
6409 00002EB5 7405      <1>      je       short vga_wp_1
6410 00002EB7 39FE      <1>      cmp      esi, edi
6411 00002EB9 72F7      <1>      jb       short vga_wp_0
6412 00002EBB C3      <1>      retn      ; nothing to do
6413                                <1> vga_wp_1:
6414 00002EBC 83C64F      <1>      add      esi, vga_memmodel - (vga_modes + 1)
6415                                <1>      ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
6416 00002EBF BF0000A00 <1>      mov      edi, 0A0000h
6417                                <1>      ;
6418 00002EC4 803E04      <1>      cmp      byte [esi], PLANAR4
6419 00002EC7 741C      <1>      je       short vga_wp_planar
6420 00002EC9 803E03      <1>      cmp      byte [esi], PLANAR1
6421 00002ECC 7417      <1>      je       short vga_wp_planar
6422                                <1> vga_wp_linear8:
6423                                <1>      ; addr=CX+DX*(read_word(BIOSMEM_SEG, BIOSMEM_NB_COLS)*8);
6424 00002ECE 0FB605[20680000] <1>      movzx   eax, byte [CRT_COLS] ; = [VGA_COLS] ; nbcols
6425                                <1>      ;shl      ax, 3 ; * 8
6426                                <1>      ; 02/08/2022
6427 00002ED5 C1E003      <1>      shl      eax, 3
6428 00002ED8 66F7E2      <1>      mul      dx
6429 00002EDB 50      <1>      push     eax
6430                                <1>      ;mov      edi, 0A0000h
6431 00002EDC 6601CF      <1>      add      di, cx
6432 00002EDF 58      <1>      pop      eax
6433 00002EE0 01C7      <1>      add      edi, eax ; addr
6434                                <1>      ; write_byte(0xa000,addr,AL);
6435 00002EE2 881F      <1>      mov      [edi], bl
6436 00002EE4 C3      <1>      retn
6437                                <1> vga_wp_planar:
6438                                <1>      ; addr = CX/8+DX*read_word(BIOSMEM_SEG, BIOSMEM_NB_COLS);
6439 00002EE5 0FB7C1      <1>      movzx   eax, cx
6440 00002EE8 66C1E803 <1>      shr      ax, 3 ; CX/8
6441 00002EEC 50      <1>      push     eax
6442 00002EED 28E4      <1>      sub      ah, ah ; 0
6443 00002EEF A0[20680000] <1>      mov      al, [CRT_COLS] ; = [VGA_COLS] ; nbcols
6444 00002EF4 66F7E2      <1>      mul      dx
6445                                <1>      ;mov      edi, 0A0000h
6446 00002EF7 6601C7      <1>      add      di, ax
6447 00002EFA 58      <1>      pop      eax
6448 00002EFB 01C7      <1>      add      edi, eax ; addr
6449 00002EFD 80E107      <1>      and      cl, 7
6450 00002F00 B580      <1>      mov      ch, 80h ; mask
6451 00002F02 D2ED      <1>      shr      ch, cl ; mask = 0x80 >> (CX & 0x07);
6452                                <1>
6453                                <1>      ; outw(VGAREG_GRDC_ADDRESS, (mask << 8) | 0x08);
6454 00002F04 66BACE03 <1>      mov      dx, 3CEh ; VGAREG_GRDC_ADDRESS
6455 00002F08 88EC      <1>      mov      ah, ch
6456 00002F0A B008      <1>      mov      al, 8
6457 00002F0C 66EF      <1>      out      dx, ax
6458                                <1>      ; outw(VGAREG_GRDC_ADDRESS, 0x0205);
6459 00002F0E 66B80502 <1>      mov      ax, 0205h
6460 00002F12 66EF      <1>      out      dx, ax
6461                                <1>      ; data = read_byte(0xa000,addr);
6462 00002F14 8A07      <1>      mov      al, [edi] ; (delay?)
6463                                <1>      ; if (AL & 0x80)
6464                                <1>      ; {
6465                                <1>      ;     outw(VGAREG_GRDC_ADDRESS, 0x1803);
6466                                <1>      ; }
6467 00002F16 F6C380      <1>      test     bl, 80h
6468 00002F19 7406      <1>      jz       short vga_wp_2
6469 00002F1B 66B80318 <1>      mov      ax, 1803h
6470 00002F1F 66EF      <1>      out      dx, ax
6471                                <1> vga_wp_2:
6472                                <1>      ; write_byte(0xa000,addr,AL);
6473 00002F21 881F      <1>      mov      [edi], bl
6474                                <1>      ;
6475                                <1>      ;mov      dx, 3CEh ; VGAREG_GRDC_ADDRESS
6476 00002F23 66B808FF <1>      mov      ax, 0FF08h
6477 00002F27 66EF      <1>      out      dx, ax
6478 00002F29 66B80500 <1>      mov      ax, 0005h
6479 00002F2D 66EF      <1>      out      dx, ax
6480 00002F2F 66B80300 <1>      mov      ax, 0003h
6481 00002F33 66EF      <1>      out      dx, ax
6482                                <1>      ;
6483 00002F35 C3      <1>      retn
6484                                <1>
6485                                <1> vga_read_pixel:
6486                                <1>      ; 02/08/2022 (TRDOS 386 kernel v2.0.5)
6487                                <1>      ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
6488                                <1>      ;
6489                                <1>      ; derived from 'Plex86/Bochs VGABios' source code
6490                                <1>      ; vgabios-0.7a (2011)
6491                                <1>      ; by the LGPL VGABios developers Team (2001-2008)
6492                                <1>      ; 'vgabios.c', 'biosfn_read_pixel'
6493                                <1>
6494                                <1>      ; INPUT ->
6495                                <1>      ; DX = row (0-239)
6496                                <1>      ; CX = column (0-799)
6497                                <1>      ; (AH = [CRT_MODE])

```

```

6498      <1>      ; OUTPUT ->
6499      <1>      ;      AL = pixel value
6500      <1>
6501      <1>      ;mov    ah, [CRT_MODE]
6502 00002F36 BE[3A680000] <1>      mov    esi, vga_modes
6503 00002F3B 89F7 <1>      mov    edi, esi
6504 00002F3D 83C710 <1>      add    edi, vga_mode_count
6505      <1> vga_rp_0:
6506      <1>      lodsb
6507 00002F41 38E0 <1>      cmp    al, ah ; [CRT_MODE]
6508 00002F43 7405 <1>      je     short vga_rp_1
6509 00002F45 39FE <1>      cmp    esi, edi
6510 00002F47 72F7 <1>      jnb    short vga_rp_0
6511 00002F49 C3 <1>      retn    ; nothing to do
6512      <1> vga_rp_1:
6513 00002F4A 83C64F <1>      add    esi, vga_memmodel - (vga_modes + 1)
6514      <1>      ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
6515 00002F4D BF00000A00 <1>      mov    edi, 0A0000h
6516      <1>      ;
6517 00002F52 803E04 <1>      cmp    byte [esi], PLANAR4
6518 00002F55 741C <1>      je     short vga_rp_planar
6519 00002F57 803E03 <1>      cmp    byte [esi], PLANAR1
6520 00002F5A 7417 <1>      je     short vga_rp_planar
6521      <1> vga_rp_linear8:
6522      <1>      ; addr=CX+DX*(read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS)*8);
6523 00002F5C 0FB605[20680000] <1>      movzx   eax, byte [CRT_COLS] ; = [VGA_COLS] ; nbcols
6524      <1>      ;shl     ax, 3 ; * 8
6525      <1>      ; 02/08/2022
6526 00002F63 C1E003 <1>      shl     eax, 3
6527 00002F66 66F7E2 <1>      mul     dx
6528 00002F69 50 <1>      push    eax
6529      <1>      ;mov    edi, 0A0000h
6530 00002F6A 6601CF <1>      add     di, cx
6531 00002F6D 58 <1>      pop     eax
6532 00002F6E 01C7 <1>      add     edi, eax ; addr
6533      <1>      ; attr=read_byte(0xa000,addr);
6534 00002F70 8A07 <1>      mov     al, [edi] ; pixel value
6535 00002F72 C3 <1>      retn
6536      <1> vga_rp_planar:
6537      <1>      ; addr = CX/8+DX*read_word(BIOSMEM_SEG,BIOSMEM_NB_COLS);
6538 00002F73 0FB7C1 <1>      movzx   eax, cx
6539 00002F76 66C1E803 <1>      shr     ax, 3 ; CX/8
6540 00002F7A 50 <1>      push    eax
6541 00002F7B 28E4 <1>      sub     ah, ah ; 0
6542 00002F7D A0[20680000] <1>      mov     al, [CRT_COLS] ; = [VGA_COLS] ; nbcols
6543 00002F82 66F7E2 <1>      mul     dx
6544      <1>      ;mov    edi, 0A0000h
6545 00002F85 6601C7 <1>      add     di, ax
6546 00002F88 58 <1>      pop     eax
6547 00002F89 01C7 <1>      add     edi, eax ; addr
6548 00002F8B 80E107 <1>      and     cl, 7
6549 00002F8E B580 <1>      mov     ch, 80h ; mask
6550 00002F90 D2ED <1>      shr     ch, cl ; mask = 0x80 >> (CX & 0x07);
6551      <1>      ; attr = 0x00;
6552 00002F92 30DB <1>      xor     bl, bl ; attr = bl = 0,
6553 00002F94 30C9 <1>      xor     cl, cl ; i = cl = 0
6554      <1>      ; for(i=0;i<4;i++)
6555      <1>      ; {
6556      <1>      ;     outw(VGAREG_GRDC_ADDRESS, (i << 8) | 0x04);
6557      <1>      ;     data = read_byte(0xa000,addr) & mask;
6558      <1>      ;     if (data > 0) attr |= (0x01 << i);
6559      <1>      ; }
6560      <1> vga_rp_2:
6561 00002F96 88CC <1>      mov     ah, cl ; i << 8
6562 00002F98 B004 <1>      mov     al, 4 ; | 0x04
6563 00002F9A 66BACE03 <1>      mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
6564 00002F9E 66EF <1>      out     dx, ax
6565      <1>      ; data = read_byte(0xa000,addr) & mask;
6566 00002FA0 8A07 <1>      mov     al, [edi]
6567 00002FA2 20E8 <1>      and     al, ch ; & mask
6568      <1>      ; if (data > 0) attr |= (0x01 << i);
6569 00002FA4 08C0 <1>      or      al, al
6570 00002FA6 7408 <1>      jz     short vga_rp_3 ; al = 0
6571 00002FA8 B701 <1>      mov     bh, 1
6572 00002FAA D2E7 <1>      shl     bh, cl ; (0x01 << i)
6573 00002FAC 08FB <1>      or      bl, bh ; attr |= (0x01 << i)
6574 00002FAE 88D8 <1>      mov     al, bl ; pixel value
6575      <1> vga_rp_3:
6576 00002FB0 C3 <1>      retn
6577      <1>
6578      <1> vga_beeper:
6579      <1>      ; 04/08/2016 (TRDOS 386 = TRDOS v2.0)
6580 00002FB1 FB <1>      sti
6581      <1>      ;mov    bh, [ACTIVE_PAGE]
6582 00002FB2 E917F4FFFF <1>      jmp     beeper_gfx
6583      <1>
6584      <1> vga_write_teletype:
6585      <1>      ; 03/08/2022 (TRDOS 386 Kernel v2.0.5)
6586      <1>      ; 12/04/2021 (TRDOS 386 v2.0.3, 32 bit push/pop)
6587      <1>      ; 09/12/2017
6588      <1>      ; 06/08/2016
6589      <1>      ; 04/08/2016
6590      <1>      ; 01/08/2016
6591      <1>      ; 31/07/2016
6592      <1>      ; 09/07/2016 (TRDOS 386 = TRDOS v2.0)
6593      <1>      ;
6594      <1>      ; derived from 'Plex86/Bochs VGABios' source code
6595      <1>      ; vgabios-0.7a (2011)
6596      <1>      ; by the LGPL VGABios developers Team (2001-2008)
6597      <1>      ; 'vgabios.c', 'biosfn_write_teletype',
6598      <1>      ; 'biosfn_write_char_only'
6599      <1>
6600      <1>      ; INPUT ->
6601      <1>      ; [CRT_MODE] = current video mode (>7)
6602      <1>      ; AL = Character to write
6603      <1>      ; BL = Color of character
6604      <1>      ; OUTPUT ->
6605      <1>      ; Regen buffer updated
6606      <1>
6607      <1>      ; biosfn_write_teletype (car, page, attr, flag)
6608      <1>      ; car = character (AL)
6609      <1>      ; page = 0
6610      <1>      ; attr = color (BL)
6611      <1>      ; 'flag' not used
6612      <1>
6613 00002FB7 8A25[1E680000] <1>      mov     ah, [CRT_MODE]
6614 00002FBD 88C7 <1>      mov     bh, al ; character
6615 00002FBF 668B15[BE760100] <1>      mov     dx, [CURSOR_POSN] ; cursor pos for page 0
6616      <1>
6617 00002FC6 BE[42680000] <1>      mov     esi, vga_g_modes
6618 00002FCB 89F7 <1>      mov     edi, esi
6619 00002FCD 83C708 <1>      add     edi, vga_g_mode_count
6620      <1> vga_wtty_0:
6621 00002FD0 AC <1>      lodsb

```

```

6622 00002FD1 38E0      <1>      cmp     al, ah ; [CRT_MODE]
6623 00002FD3 7405      <1>      je      short vga_wtty_2
6624 00002FD5 39FE      <1>      cmp     esi, edi
6625 00002FD7 72F7      <1>      jb      short vga_wtty_0
6626                                     <1> vga_wtty_1:
6627 00002FD9 C3          <1>      retn    ; nothing to do
6628                                     <1> vga_wtty_2:
6629 00002FDA 80FF07     <1>      cmp     bh, 07h ; bell (beep)
6630 00002FDD 74D2      <1>      je      short vga_beeper ; u11
6631 00002FDF 80FF08     <1>      cmp     bh, 08h ; backspace
6632 00002FE2 7508      <1>      jne     short vga_wtty_3
6633                                     <1>      ; if(xcurs>0)xcurs--;
6634 00002FE4 08D2      <1>      or      dl, dl ; xcurs (column)
6635 00002FE6 74F1      <1>      jz      short vga_wtty_1
6636 00002FE8 FECA      <1>      dec     dl ; xcurs--;
6637 00002FEA EB55      <1>      jmp     short vga_wtty_12
6638                                     <1> vga_wtty_3:
6639 00002FEC 80FF0D     <1>      cmp     bh, 0Dh ; carriage return (\r)
6640 00002FEF 7504      <1>      jne     short vga_wtty_4
6641                                     <1>      ; xcurs=0;
6642 00002FF1 28D2      <1>      sub     dl, dl ; 0
6643 00002FF3 EB4C      <1>      jmp     short vga_wtty_12
6644                                     <1> vga_wtty_4:
6645 00002FF5 80FF0A     <1>      cmp     bh, 0Ah ; new line (\n)
6646 00002FF8 7504      <1>      jne     short vga_wtty_5
6647                                     <1>      ; ycurs++;
6648 00002FFA FEC6      <1>      inc     dh ; next row
6649 00002FFC EB5E      <1>      jmp     short vga_wtty_11
6650                                     <1> vga_wtty_5:
6651 00002FFE 80FF09     <1>      cmp     bh, 09h ; tab stop
6652 00003001 7523      <1>      jne     short vga_wtty_8
6653 00003003 88D0      <1>      mov     al, dl
6654                                     <1>      ;cbw
6655 00003005 30E4      <1>      xor     ah, ah ; 09/12/2017
6656 00003007 B108      <1>      mov     cl, 8
6657 00003009 F6F1      <1>      div     cl
6658 0000300B 28E1      <1>      sub     cl, ah
6659                                     <1>      ;
6660 0000300D B720      <1>      mov     bh, 20h ; space
6661                                     <1> vga_wtty_6: ; tab stop loop
6662                                     <1>      ;push cx
6663                                     <1>      ;push bx
6664                                     <1>      ; 12/04/2021
6665 0000300F 51          <1>      push    ecx
6666 00003010 53          <1>      push    ebx
6667 00003011 E810000000 <1>      call    vga_wtty_8
6668                                     <1>      ;pop bx ; bh = character, bl = color
6669                                     <1>      ;pop cx
6670                                     <1>      ; 12/04/2021
6671 00003016 5B          <1>      pop     ebx ; bh = character, bl = color
6672 00003017 59          <1>      pop     ecx
6673 00003018 FEC9      <1>      dec     cl
6674 0000301A 7409      <1>      jz      short vga_wtty_7
6675 0000301C 668B15[BE760100] <1>      mov     dx, [CURSOR_POSN] ; new cursor position (pg 0)
6676 00003023 EBEA      <1>      jmp     short vga_wtty_6
6677                                     <1> vga_wtty_7:
6678 00003025 C3          <1>      retn
6679                                     <1>      ;
6680                                     <1> vga_wtty_8:
6681 00003026 83C64F     <1>      add     esi, vga_g_memmodel - (vga_g_modes + 1)
6682                                     <1>      ; [ESI] = VGA memory model number (LINEAR8, PLANAR4, PLANAR1)
6683 00003029 BF00000A00 <1>      mov     edi, 0A0000h
6684                                     <1>      ;
6685 0000302E 88F8      <1>      mov     al, bh ; character
6686                                     <1>      ;
6687 00003030 803E04     <1>      cmp     byte [esi], PLANAR4
6688 00003033 7414      <1>      je      short vga_wtty_planar
6689 00003035 803E03     <1>      cmp     byte [esi], PLANAR1
6690 00003038 740F      <1>      je      short vga_wtty_planar
6691                                     <1> vga_wtty_linear8:
6692                                     <1>      ; write_gfx_char_lin(car,attr,xcurs,ycurs,nbcols);
6693                                     <1>      ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,
6694                                     <1>      ; [CRT_COLS] = nbcols
6695 0000303A E8F1FCFFFF <1>      call    write_gfx_char_lin
6696 0000303F EB0D      <1>      jmp     short vga_wtty_9
6697                                     <1>      ;
6698                                     <1> vga_wtty_12:
6699                                     <1>      ; 09/07/2016
6700                                     <1>      ; set cursor position
6701                                     <1>      ; NOTE: Hardware cursor position will not be set
6702                                     <1>      ; in any VGA modes (>7)
6703                                     <1>      ; But, cursor position will be saved into
6704                                     <1>      ; [CURSOR_POSN].
6705                                     <1>      ; TRDOS 386 (TRDOS v2.0) uses only one page
6706                                     <1>      ; (page 0) for all graphics modes.
6707                                     <1>      ;
6708 00003041 668915[BE760100] <1>      mov     [CURSOR_POSN], dx ; save cursor pos for pg 0
6709                                     <1>      ; 04/08/2016
6710                                     <1>      ;mov bh, [ACTIVE_PAGE] ; = 0
6711                                     <1>      ;call _set_cpos
6712 00003048 C3          <1>      retn
6713                                     <1>      ;
6714                                     <1> vga_wtty_planar:
6715                                     <1>      ; write_gfx_char_pl4(car,attr,xcurs,ycurs,nbcols,cheight);
6716                                     <1>      ; AL = car, BL = attr (color), DL = xcurs, DH = ycurs,
6717                                     <1>      ; [CRT_COLS]= nbcols, [CHAR_HEIGHT] = cheight
6718 00003049 E883FDFFFF <1>      call    write_gfx_char_pl4
6719                                     <1> vga_wtty_9:
6720 0000304E FEC2      <1>      inc     dl ; xcurs++;
6721                                     <1> vga_wtty_10:
6722                                     <1>      ; Do we need to wrap ?
6723                                     <1>      ; if(xcurs==nbcols)
6724 00003050 3A15[20680000] <1>      cmp     dl, [CRT_COLS] ; [VGA_COLS]
6725 00003056 7204      <1>      jb      short vga_wtty_11 ; no
6726 00003058 28D2      <1>      sub     dl, dl ; xcurs=0;
6727 0000305A FEC6      <1>      inc     dh ; ycurs++;
6728                                     <1> vga_wtty_11:
6729                                     <1>      ; Do we need to scroll ?
6730                                     <1>      ; if(ycurs==nbrows)
6731 0000305C 3A35[26680000] <1>      cmp     dh, [VGA_ROWS]
6732 00003062 72DD      <1>      jb      short vga_wtty_12 ; no
6733                                     <1>      ;
6734                                     <1>      ; biosfn_scroll (nblines,attr,rul,cu1,r1r,clr,page,dir)
6735                                     <1>      ; al = nblines = 1, bl = attr (color) = 0
6736                                     <1>      ; ch = rul, cl = cu1, dh = r1r, dl = clr, page = 0
6737                                     <1>      ; dir = SCROLL_UP
6738                                     <1>      ;
6739 00003064 B001      <1>      mov     al, 1
6740 00003066 28DB      <1>      sub     bl, bl ; 0 ; blank/black line (attr=0) will be used
6741                                     <1>      ;sub cx, cx ; 0,0
6742                                     <1>      ; 03/08/2022
6743 00003068 29C9      <1>      sub     ecx, ecx
6744                                     <1>      ;
6745                                     <1>      ; 06/08/2016

```



```

6746 0000306A 8A35[26680000] <1> mov dh, [VGA_ROWS]
6747 00003070 FECE <1> dec dh ; nbrows -1
6748 <1>
6749 <1> ;push dx ; 04/08/2016
6750 <1> ; 12/04/2021
6751 00003072 52 <1> push edx
6752 00003073 8A15[20680000] <1> mov dl, [CRT_COLS]
6753 00003079 FECA <1> dec dl ; nbcols -1
6754 <1>
6755 0000307B 8A25[1E680000] <1> mov ah, [CRT_MODE]
6756 <1>
6757 <1> ; biosfn_scroll(0x01,0x00,0,0,nbrows-1,nbcols-1,page,SCROLL_UP);
6758 00003081 E843F5FFFF <1> call vga_graphics_up
6759 <1> ; 04/08/2016
6760 <1> ;pop dx
6761 <1> ; 12/04/2021
6762 00003086 5A <1> pop edx
6763 <1>
6764 <1> ;dec dh ; ycurs-=1
6765 00003087 EBB8 <1> jmp short vga_wtty_12
6766 <1>
6767 <1> font_setup:
6768 <1> ; 03/08/2022 (TRDOS 386 v2.0.5)
6769 <1> ; 09/01/2021 (TRDOS 386 v2.0.3)
6770 <1> ; 09/07/2016
6771 <1> ; character generator (font loading) functions
6772 <1> ;
6773 <1> ; derived from 'Plex86/Bochs VGABios' source code
6774 <1> ; vgabios-0.7a (2011)
6775 <1> ; by the LGPL VGABios developers Team (2001-2008)
6776 <1> ; 'vgabios.c', 'int10_func'
6777 <1>
6778 <1> ; AX = 1100H ; Load User-Defined Font (EGA/VGA)
6779 <1> ;
6780 <1> ; BH = height of each character (bytes per character definition)
6781 <1> ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
6782 <1> ; CX = number of characters to redefine (<=256)
6783 <1> ; DX = ASCII code of the first character defined at ES:BP
6784 <1> ; EBP = address of font-definition information
6785 <1> ; (in user's memory space)
6786 <1>
6787 <1> ; case 0x11:
6788 <1> ; switch(GET_AL())
6789 <1> ; {
6790 <1> ; case 0x00:
6791 <1> ; case 0x10:
6792 <1> ; biosfn_load_text_user_pat(GET_AL(),ES,BP,CX,DX,GET_BL(),GET_BH());
6793 <1> ; break;
6794 <1>
6795 <1> ; AX = 1110H ; Load and Activate User-Defined Font (EGA/VGA)
6796 00003089 08C0 <1> or al, al ; 0
6797 0000308B 7404 <1> jz short font_setup_0
6798 0000308D 3C10 <1> cmp al, 10h
6799 0000308F 7511 <1> jne short font_setup_1
6800 <1> font_setup_0:
6801 00003091 E8CE000000 <1> call transfer_user_fonts
6802 00003096 721C <1> jc short font_setup_error
6803 00003098 E8AD010000 <1> call load_text_user_pat
6804 0000309D E9D6EAF5FF <1> jmp VIDEO_RETURN
6805 <1> font_setup_1:
6806 <1> ; AX = 1101H ; Load ROM 8x14 Character Set (EGA/VGA)
6807 <1> ; case 0x01:
6808 <1> ; case 0x11:
6809 <1> ; biosfn_load_text_8_14_pat(GET_AL(),GET_BL());
6810 <1> ; break;
6811 000030A2 3C01 <1> cmp al, 1
6812 000030A4 7404 <1> je short font_setup_2
6813 000030A6 3C11 <1> cmp al, 11h
6814 000030A8 7511 <1> jne short font_setup_3
6815 <1> font_setup_2:
6816 <1> ; AX = 1111H ; Load and Activate ROM 8x14 Character Set (EGA/VGA)
6817 <1> ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
6818 000030AA E8C9020000 <1> call load_text_8_14_pat
6819 000030AF E9C4EAF5FF <1> jmp VIDEO_RETURN
6820 <1> font_setup_error:
6821 000030B4 29C0 <1> sub eax, eax ; 0 -> fonts could not be loaded
6822 000030B6 E9C2EAF5FF <1> jmp _video_return
6823 <1> font_setup_3:
6824 <1> ; AX = 1102H ; Load ROM 8x8 Character Set (EGA/VGA)
6825 <1> ; case 0x02:
6826 <1> ; case 0x12:
6827 <1> ; biosfn_load_text_8_8_pat(GET_AL(),GET_BL());
6828 <1> ; break;
6829 000030BB 3C02 <1> cmp al, 2
6830 000030BD 7404 <1> je short font_setup_4
6831 000030BF 3C12 <1> cmp al, 12h
6832 000030C1 750A <1> jne short font_setup_5
6833 <1> font_setup_4:
6834 <1> ; AX = 1112H ; Load and Activate ROM 8x8 Character Set (EGA/VGA)
6835 <1> ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
6836 000030C3 E8E1020000 <1> call load_text_8_8_pat
6837 000030C8 E9ABEAF5FF <1> jmp VIDEO_RETURN
6838 <1> font_setup_5:
6839 <1> ; AX = 1104H ; Load ROM 8x16 Character Set (EGA/VGA)
6840 <1> ; case 0x04:
6841 <1> ; case 0x14:
6842 <1> ; biosfn_load_text_8_16_pat(GET_AL(),GET_BL());
6843 <1> ; break;
6844 000030CD 3C04 <1> cmp al, 4
6845 000030CF 7404 <1> je short font_setup_6
6846 000030D1 3C14 <1> cmp al, 14h
6847 000030D3 750A <1> jne short font_setup_7
6848 <1> font_setup_6:
6849 <1> ; AX = 1114H ; Load and Activate ROM 8x16 Character Set (EGA/VGA)
6850 <1> ; (BL = font block to load (EGA: 0-3; VGA: 0-7))
6851 000030D5 E819030000 <1> call load_text_8_16_pat
6852 000030DA E999EAF5FF <1> jmp VIDEO_RETURN
6853 <1> font_setup_7:
6854 <1> ; Note: AX=1120h (Setup INT 1Fh, EXT_PTR) is not needed
6855 <1> ; for TRDOS 386 (TRDOS v2.0) video functionality;
6856 <1> ; because, originally EXT_PTR (font address) was used for
6857 <1> ; chars 80h to 0FFh (after the first 128 ASCII char fonts), for
6858 <1> ; CGA graphics mode; currently, 'vgafont8' address has 256 chars!
6859 <1> ;
6860 <1> ; case 0x20:
6861 <1> ; biosfn_load_gfx_8_8_chars(ES,BP);
6862 <1> ; break;
6863 <1> ; case 0x21:
6864 <1> ; biosfn_load_gfx_user_chars(ES,BP,CX,GET_BL(),GET_DL());
6865 <1> ; break;
6866 <1> ; AX = 1121H ; Setup User-Defined Font for Graphics Mode (VGA)
6867 <1> ; BL screen rows code: 00H = user-specified (in DL)
6868 <1> ; 01H = 14 rows
6869 <1> ; 02H = 25 rows

```

```

6870      <1>      ;                                03H = 43 rows
6871      <1>      ; CX    bytes per character definition
6872      <1>      ; DL    (when BL=0) custom number of character rows on screen
6873      <1>      ; EBP   address of font-definition information (user's mem space)
6874      <1>
6875      000030DF 3C21      <1>      cmp     al, 21h
6876      000030E1 7531      <1>      jne     short font_setup_9
6877      <1>
6878      <1>      ; TRDOS 386 modification !
6879      <1>      ; dh = 0 -> 256 characters
6880      <1>      ; dh = 80h -> second 128 characters
6881      <1>      ; dh = 0FFh -> first 128 characters
6882      <1>
6883      <1>      ; 09/01/2021 (TRDOS 386 v2.0.3)
6884      <1>      ;push    ebx
6885      000030E3 51        <1>      push    ecx
6886      000030E4 52        <1>      push    edx
6887      000030E5 30D2      <1>      xor     dl, dl
6888      000030E7 88CF      <1>      mov     bh, cl ; character height
6889      <1>      ;mov     cx, 100h ; 256
6890      <1>      ; 03/08/2022
6891      000030E9 31C9      <1>      xor     ecx, ecx
6892      000030EB FEC5      <1>      inc     ch
6893      <1>      ; ecx = 100h
6894      000030ED 08F6      <1>      or      dh, dh ; 0
6895      000030EF 7410      <1>      jz      short font_setup_8
6896      000030F1 FECD      <1>      dec     ch ; cx = 0
6897      000030F3 80FEFF      <1>      cmp     dh, 0FFh
6898      000030F6 7409      <1>      je      short font_setup_8 ; 1st 128 chars
6899      <1>      ; 2nd 128 chars
6900      000030F8 80FE80      <1>      cmp     dh, 80h
6901      000030FB 75B7      <1>      jne     short font_setup_error ; invalid !
6902      000030FD 88F1      <1>      mov     cl, dh
6903      000030FF 86F2      <1>      xchg    dl, dh
6904      <1>      ; number of chars, cx = 80h
6905      <1>      ; start char, dl = 80h
6906      <1>      font_setup_8:
6907      00003101 E85E000000 <1>      call    transfer_user_fonts
6908      00003106 5A        <1>      pop     edx
6909      00003107 59        <1>      pop     ecx
6910      <1>      ;pop    ebx
6911      00003108 72AA      <1>      jc      short font_setup_error
6912      <1>      ; ebp = user's font data address in system's memory space
6913      0000310A E82F030000 <1>      call    load_gfx_user_chars
6914      0000310F E964EAFFFF <1>      jmp     VIDEO_RETURN
6915      <1>      font_setup_9:
6916      <1>      ; case 0x22:
6917      <1>      ; biosfn_load_gfx_8_14_chars(GET_BL());
6918      <1>      ; break;
6919      00003114 3C22      <1>      cmp     al, 22h
6920      00003116 750A      <1>      jne     short font_setup_10
6921      00003118 E85D030000 <1>      call    load_gfx_8_14_chars
6922      0000311D E956EAFFFF <1>      jmp     VIDEO_RETURN
6923      <1>      font_setup_10:
6924      <1>      ; case 0x23:
6925      <1>      ; biosfn_load_gfx_8_8_dd_chars(GET_BL());
6926      <1>      ; break;
6927      00003122 3C23      <1>      cmp     al, 23h
6928      00003124 750A      <1>      jne     short font_setup_11
6929      00003126 E890030000 <1>      call    load_gfx_8_8_chars
6930      0000312B E948EAFFFF <1>      jmp     VIDEO_RETURN
6931      <1>      font_setup_11:
6932      <1>      ; case 0x24:
6933      <1>      ; biosfn_load_gfx_8_16_chars(GET_BL());
6934      <1>      ; break;
6935      00003130 3C24      <1>      cmp     al, 24h
6936      00003132 750A      <1>      jne     short font_setup_12
6937      00003134 E8C3030000 <1>      call    load_gfx_8_16_chars
6938      00003139 E93AEAFFFF <1>      jmp     VIDEO_RETURN
6939      <1>      font_setup_12:
6940      <1>      ; case 0x30:
6941      <1>      ; biosfn_get_font_info(GET_BH(), &ES, &BP, &CX, &DX);
6942      <1>      ; break;
6943      0000313E 3C30      <1>      cmp     al, 30h
6944      00003140 750A      <1>      jne     short font_setup_13
6945      00003142 E8F6030000 <1>      call    get_font_info
6946      <1>      ; eax = return value (info: 4 bytes for 4 parms)
6947      <1>      ; eax = 0 -> invalid function (input)
6948      00003147 E931EAFFFF <1>      jmp     _video_return
6949      <1>      font_setup_13:
6950      <1>      cmp     al, 03h ; AX = 1103h
6951      0000314E 750D      <1>      jne     short font_setup_14
6952      <1>      ; biosfn_set_text_block_specifier:
6953      <1>      ; BL = font block selector code
6954      <1>      ; NOTE: TRDOS 386 only uses and sets font block 0
6955      <1>      ; (It is as BL = 0 for TRDOS 386)
6956      00003150 66BAC403 <1>      mov     dx, 3C4h ; VGAREG_SEQU_ADDRESS
6957      <1>      ;mov     ah, bl
6958      00003154 28E4      <1>      sub     ah, ah ; 0
6959      <1>      ;mov     al, 03h
6960      00003156 66EF      <1>      out     dx, ax
6961      00003158 E91BEAFFFF <1>      jmp     VIDEO_RETURN
6962      <1>
6963      <1>      font_setup_14:
6964      0000315D 29C0      <1>      sub     eax, eax ; 0 = invalid function
6965      0000315F E919EAFFFF <1>      jmp     _video_return
6966      <1>
6967      <1>      transfer_user_fonts:
6968      <1>      ; 02/08/2022 (TRDOS 386 kernel v2.0.5)
6969      <1>      ; 19/01/2021
6970      <1>      ; 09/01/2021
6971      <1>      ; 05/01/2021 (TRDOS 386 v2.0.3)
6972      <1>
6973      <1>      ; BH = height of each character (bytes per character)
6974      <1>      ; CX = number of characters to redefine (<=256)
6975      <1>      ; DX = ASCII code of the first character defined at EBP
6976      <1>      ; EBP = address of font-definition information
6977      <1>      ; (in user's memory space)
6978      <1>
6979      <1>      ; Modified registers: eax, edx, ecx, esi, edi, ebp
6980      <1>      ;
6981      <1>      ; output:
6982      <1>      ; ebp = user font data address in system memory
6983      <1>
6984      00003164 81E2FFFF0000 <1>      and     edx, 0FFFFh
6985      0000316A 81E1FFFF0000 <1>      and     ecx, 0FFFFh
6986      00003170 7537      <1>      jnz     short transfer_user_fonts_5
6987      <1>
6988      <1>      or      edx, edx
6989      00003174 7531      <1>      jnz     short transfer_user_fonts_4
6990      00003176 09ED      <1>      or      ebp, ebp
6991      00003178 752D      <1>      jnz     short transfer_user_fonts_4
6992      <1>
6993      <1>      ; cx = 0, dx = 0, ebp = 0

```

```

6994      <1>      ; copy system font to user font
6995      <1>
6996      0000317A B140      <1>      mov     cl, 64 ; 64 dwords
6997      <1>
6998      0000317C 80FF10      <1>      cmp     bh, 16
6999      0000317F 7417      <1>      je      short transfer_user_fonts_2
7000      00003181 80FF08      <1>      cmp     bh, 8
7001      00003184 7406      <1>      je      short transfer_user_fonts_1
7002      <1>
7003      00003186 BD[30550100] <1>      mov     ebp, vgafont14
7004      0000318B C3      <1>      retn
7005      <1>
7006      <1> transfer_user_fonts_1:
7007      0000318C BF00500900 <1>      mov     edi, VGAFONT8USER
7008      00003191 BE[304D0100] <1>      mov     esi, vgafont8
7009      00003196 EB0A      <1>      jmp     short transfer_user_fonts_3
7010      <1>
7011      <1> transfer_user_fonts_2:
7012      00003198 BF00400900 <1>      mov     edi, VGAFONT16USER
7013      0000319D BE[30630100] <1>      mov     esi, vgafont16
7014      <1> transfer_user_fonts_3:
7015      000031A2 89FD      <1>      mov     ebp, edi
7016      000031A4 F3A5      <1>      rep     movsd
7017      000031A6 C3      <1>      retn
7018      <1>
7019      <1> transfer_user_fonts_4:
7020      000031A7 F9      <1>      stc
7021      000031A8 C3      <1>      retn
7022      <1>
7023      <1> transfer_user_fonts_5:
7024      000031A9 09ED      <1>      or      ebp, ebp
7025      000031AB 74FA      <1>      jz      short transfer_user_fonts_4 ; invalid address !
7026      <1>
7027      000031AD 6681F90001 <1>      cmp     cx, 256
7028      000031B2 77F3      <1>      ja      short transfer_user_fonts_4
7029      000031B4 29D1      <1>      sub     ecx, edx
7030      000031B6 76EF      <1>      jna     short transfer_user_fonts_4
7031      <1>
7032      000031B8 80FF0E      <1>      cmp     bh, 14 ; 8x14 font
7033      <1>      ; (there is not an alternative buffer)
7034      000031BB 7524      <1>      jne     short transfer_user_fonts_6
7035      <1>
7036      <1> ; use system's 8x14 font space if permission flag is 1
7037      000031BD F605[6E9D0100]80 <1>      test    byte [ufont], 80h
7038      000031C4 74E1      <1>      jz      short transfer_user_fonts_4 ; not allowed
7039      <1>
7040      <1> ; permission is given (for vgafont14 location etc.)
7041      <1> ; (for permanent font modification)
7042      <1> ;
7043      <1> ; 19/01/2021
7044      <1> ; Note: Permission flag can be set by 'root' while
7045      <1> ; system is not in multi tasking/user mode
7046      <1> ; while [multi_tasking] = 0 and [u.uid] = 0
7047      <1>
7048      <1> ;push  edx
7049      <1> ; 02/08/2022
7050      000031C6 87CA      <1>      xchg    edx, ecx
7051      <1> ;xor    ah, ah
7052      <1> ; 02/08/2022
7053      000031C8 31C0      <1>      xor     eax, eax
7054      000031CA 88F8      <1>      mov     al, bh ; mov al, 14
7055      <1> ;mul    cx
7056      <1> ; 02/08/2022
7057      000031CC F7E2      <1>      mul     edx ; char count * 14
7058      <1> ; 02/08/2022
7059      000031CE 89CA      <1>      mov     edx, ecx ; ascii code
7060      000031D0 89C1      <1>      mov     ecx, eax
7061      <1> ; ecx = byte count
7062      <1> ;pop     edx
7063      <1> ; 02/08/2022
7064      <1> ;xor     eax, eax
7065      000031D2 30E4      <1>      xor     ah, ah
7066      000031D4 88F8      <1>      mov     al, bh ; mov ax, 14 ; bytes per character
7067      <1> ;mul    dx
7068      <1> ;mov    dx, ax ; char offset
7069      <1> ; 02/08/2022
7070      000031D6 F7E2      <1>      mul     edx
7071      000031D8 89C2      <1>      mov     edx, eax ; char offset
7072      000031DA BF[30550100] <1>      mov     edi, vgafont14
7073      000031DF EB48      <1>      jmp     short transfer_user_fonts_8
7074      <1> transfer_user_fonts_6:
7075      000031E1 80FF08      <1>      cmp     bh, 8 ; 8x8 font
7076      000031E4 7520      <1>      jne     short transfer_user_fonts_7 ; 8x16 font
7077      <1> ;shl    dx, 3 ; * 8
7078      <1> ;shl    cx, 3 ; * 8
7079      <1> ; 02/08/2022
7080      000031E6 C1E203      <1>      shl     edx, 3 ; byte offset
7081      000031E9 C1E103      <1>      shl     ecx, 3 ; byte count
7082      <1> ; 09/01/2021
7083      000031EC BF00500900 <1>      mov     edi, VGAFONT8USER
7084      000031F1 F605[6E9D0100]08 <1>      test    byte [ufont], 8 ; already loaded ?
7085      000031F8 752F      <1>      jnz     short transfer_user_fonts_8 ; yes
7086      000031FA BE[304D0100] <1>      mov     esi, vgafont8
7087      000031FF E839000000 <1>      call    transfer_user_fonts_10
7088      00003204 EB23      <1>      jmp     short transfer_user_fonts_8
7089      <1> transfer_user_fonts_7:
7090      00003206 80FF10      <1>      cmp     bh, 16 ; 8x16 font
7091      00003209 759C      <1>      jne     short transfer_user_fonts_4 ; invalid !
7092      <1> ;shl    dx, 4 ; * 16
7093      <1> ;shl    cx, 4 ; * 16
7094      <1> ; 02/08/2022
7095      0000320B C1E204      <1>      shl     edx, 4 ; byte offset
7096      0000320E C1E104      <1>      shl     ecx, 4 ; byte count
7097      00003211 BF00400900 <1>      mov     edi, VGAFONT16USER
7098      00003216 F605[6E9D0100]10 <1>      test    byte [ufont], 16 ; already loaded ?
7099      0000321D 750A      <1>      jnz     short transfer_user_fonts_8 ; yes
7100      0000321F BE[30630100] <1>      mov     esi, vgafont16
7101      00003224 E814000000 <1>      call    transfer_user_fonts_10
7102      <1> transfer_user_fonts_8:
7103      00003229 01D7      <1>      add     edi, edx ; char offset
7104      <1> ; 09/07/2016
7105      <1> ;and    ecx, 0FFFFh
7106      <1> ; ECX = byte count
7107      <1> ;push    ecx
7108      0000322B 89EE      <1>      mov     esi, ebp ; user's font buffer
7109      <1> ; 09/01/2021
7110      0000322D 89FD      <1>      mov     ebp, edi ; system addr for user's font
7111      <1> ; 05/01/2021
7112      <1> ;mov     edi, Cluster_Buffer ; system buffer
7113      0000322F E821D90000 <1>      call    transfer_from_user_buffer
7114      <1> ;pop     ecx
7115      <1> ; ecx = transfer (byte) count = character count
7116      00003234 7206      <1>      jc      short transfer_user_fonts_9
7117      <1> ; 05/01/2021

```

```

7118      ;mov     ebp, cluster_Buffer
7119
7120 00003236 083D[6E9D0100]      or      byte [ufont], bh
7121      ; 8x8 or 8x16 user font ready
7122      transfer_user_fonts_9:
7123 0000323C C3                  retn
7124
7125      transfer_user_fonts_10:
7126      ; 02/08/2022
7127      ; 09/01/2021
7128 0000323D 56                  push    esi
7129 0000323E 57                  push    edi
7130 0000323F 51                  push    ecx
7131      ;mov     cx, 64
7132      ; 02/08/2022
7133 00003240 31C9              xor     ecx, ecx
7134 00003242 B140              mov     cl, 64
7135 00003244 F3A5              rep     movsd
7136 00003246 59                  pop     ecx
7137 00003247 5F                  pop     edi
7138 00003248 5E                  pop     esi
7139 00003249 C3                  retn
7140
7141      load_text_user_pat:
7142      ; 02/08/2022 (TRDOS 3386 v2.0.5)
7143      ; 26/07/2016
7144      ; 09/07/2016
7145      ; load user defined (EGA/VGA) text fonts
7146      ;
7147      ; derived from 'Plex86/Bochs VGABios' source code
7148      ; vgabios-0.7a (2011)
7149      ; by the LGPL VGABios developers Team (2001-2008)
7150      ; 'vgabios.c', 'biosfn_load_text_user_pat'
7151
7152      ; biosfn_load_text_user_pat (AL,ES,BP,CX,DX,BL,BH)
7153
7154      ; get_font_access();
7155      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
7156      ; for(i=0;i<CX;i++)
7157      ; {
7158      ;     src = BP + i * BH;
7159      ;     dest = blockaddr + (DX + i) * 32;
7160      ;     memcpyb(0xA000, dest, ES, src, BH);
7161      ; }
7162      ; release_font_access();
7163      ; if(AL>=0x10)
7164      ; {
7165      ;     set_scan_lines(BH);
7166      ; }
7167
7168 0000324A 50                  push    eax
7169 0000324B E839000000          call    get_font_access
7170 00003250 28DB              sub     b1, b1 ; i = 0
7171      ; 02/08/2022
7172      ; ecx <= 256
7173 00003252 30ED              xor     ch, ch
7174      ltup_1:
7175 00003254 88D8              mov     al, b1
7176 00003256 F6E7              mul     bh
7177      ;movzx    esi, ax
7178      ; 02/08/2022
7179 00003258 89C6              mov     esi, eax
7180 0000325A 01EE              add     esi, ebp
7181 0000325C 88D8              mov     al, b1
7182 0000325E 28E4              sub     ah, ah
7183      ;add     ax, dx ; (DX + i)
7184      ;shl     ax, 5 ; * 32
7185      ; 02/08/2022
7186 00003260 01D0              add     eax, edx
7187 00003262 C1E005          shl     eax, 5
7188      ;movzx    edi, ax
7189      ; 02/08/2022
7190 00003265 89C7              mov     edi, eax
7191 00003267 81C700000A00          add     edi, 0A0000h
7192 0000326D 51                  push    ecx
7193      ;movzx    ecx, bh
7194      ; 02/08/2022
7195 0000326E 88F9              mov     cl, bh
7196 00003270 F3A4              rep     movsb
7197 00003272 59                  pop     ecx
7198 00003273 FEC3              inc     b1
7199 00003275 38CB              cmp     b1, cl
7200 00003277 75DB              jne     short ltup_1
7201      ;
7202 00003279 E83E000000          call    release_font_access
7203      ;
7204 0000327E 58                  pop     eax
7205      ; if(AL>=0x10)
7206 0000327F 3C10              cmp     al, 10h
7207 00003281 7205              jb     short ltup_2
7208      ; set_scan_lines(BH);
7209 00003283 E86F000000          call    set_scan_lines
7210      ltup_2:
7211 00003288 C3                  retn
7212
7213      get_font_access:
7214      ; 02/08/2022
7215      ; 09/07/2016
7216      ;
7217      ; derived from 'Plex86/Bochs VGABios' source code
7218      ; vgabios-0.7a (2011)
7219      ; by the LGPL VGABios developers Team (2001-2008)
7220      ; 'vgabios.c', 'get_font_access'
7221
7222      ; get_font_access()
7223 00003289 52                  push    edx
7224 0000328A 66BAC403          mov     dx, 3C4h ; VGAREG_SEQU_ADDRESS
7225      ; 02/08/2022
7226 0000328E 31C0              xor     eax, eax
7227      ;mov     ax, 0100h
7228 00003290 B401              mov     ah, 1
7229      ; ax = 0100h
7230 00003292 66EF              out     dx, ax
7231 00003294 66B80204          mov     ax, 0402h
7232 00003298 66EF              out     dx, ax
7233 0000329A 66B80407          mov     ax, 0704h
7234 0000329E 66EF              out     dx, ax
7235 000032A0 66B80003          mov     ax, 0300h
7236 000032A4 66EF              out     dx, ax
7237      ;mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
7238      ; 02/08/2022
7239 000032A6 B2CE              mov     dl, 0CEh
7240 000032A8 66B80402          mov     ax, 0204h
7241 000032AC 66EF              out     dx, ax

```

```

7242 000032AE 66B80500 <1> mov ax, 0005h
7243 000032B2 66EF <1> out dx, ax
7244 000032B4 66B80604 <1> mov ax, 0406h
7245 000032B8 66EF <1> out dx, ax
7246 000032BA 5A <1> pop edx
7247 000032BB C3 <1> retn
7248 <1>
7249 <1> release_font_access:
7250 <1> ; 02/08/2022
7251 <1> ; 29/07/2016
7252 <1> ; 09/07/2016
7253 <1> ;
7254 <1> ; derived from 'Plex86/Bochs VGABios' source code
7255 <1> ; vgabios-0.7a (2011)
7256 <1> ; by the LGPL VGABios developers Team (2001-2008)
7257 <1> ; 'vgabios.c', 'release_font_access'
7258 <1>
7259 000032BC 66BAC403 <1> mov dx, 3C4h ; VGAREG_SEQU_ADDRESS
7260 <1> ;mov ax, 0100h
7261 <1> ; 02/08/2022
7262 000032C0 29C0 <1> sub eax, eax
7263 000032C2 B401 <1> mov ah, 1
7264 <1> ; ax = 0100h
7265 000032C4 66EF <1> out dx, ax
7266 000032C6 66B80203 <1> mov ax, 0302h
7267 000032CA 66EF <1> out dx, ax
7268 000032CC 66B80403 <1> mov ax, 0304h
7269 000032D0 66EF <1> out dx, ax
7270 000032D2 66B80003 <1> mov ax, 0300h
7271 000032D6 66EF <1> out dx, ax
7272 <1> ;mov dx, 3CCh ; VGAREG_READ_MISC_OUTPUT
7273 <1> ; 02/08/2022
7274 000032D8 B2CC <1> mov dl, 0Cch
7275 000032DA EC <1> in al, dx
7276 000032DB 2401 <1> and al, 01h
7277 000032DD C0E002 <1> shl al, 2
7278 000032E0 0C0A <1> or al, 0Ah
7279 000032E2 88C4 <1> mov ah, al
7280 000032E4 B006 <1> mov al, 06h
7281 <1> ;mov dx, 3CEh ; VGAREG_GRDC_ADDRESS
7282 <1> ; 02/08/2022
7283 000032E6 B2CE <1> mov dl, 0CEh
7284 000032E8 66EF <1> out dx, ax
7285 000032EA 66B80400 <1> mov ax, 0004h
7286 000032EE 66EF <1> out dx, ax
7287 000032F0 66B80510 <1> mov ax, 1005h
7288 000032F4 66EF <1> out dx, ax
7289 000032F6 C3 <1> retn
7290 <1>
7291 <1> set_scan_lines:
7292 <1> ; 02/08/2022
7293 <1> ; 09/07/2016
7294 <1> ;
7295 <1> ; derived from 'Plex86/Bochs VGABios' source code
7296 <1> ; vgabios-0.7a (2011)
7297 <1> ; by the LGPL VGABios developers Team (2001-2008)
7298 <1> ; 'vgabios.c', 'set_scan_lines'
7299 <1>
7300 <1> ; set_scan_lines(lines)
7301 <1> ; BH = lines
7302 <1>
7303 <1> ; outb(crtc_addr, 0x09);
7304 000032F7 66BAD403 <1> mov dx, 3D4h ; CRTC_ADDRESS = 3D4h (always)
7305 000032FB B009 <1> mov al, 09h
7306 000032FD EE <1> out dx, al
7307 <1> ; crtc_r9 = inb(crtc_addr+1);
7308 <1> ;inc dx ; 3D5h
7309 <1> ; 02/08/2022
7310 000032FE FEC2 <1> inc dl
7311 00003300 EC <1> in al, dx
7312 <1> ; crtc_r9 = (crtc_r9 & 0xe0) | (lines - 1);
7313 00003301 24E0 <1> and al, 0E0h
7314 00003303 FECF <1> dec bh ; lines - 1
7315 00003305 08F8 <1> or al, bh
7316 <1> ; outb(crtc_addr+1, crtc_r9);
7317 00003307 EE <1> out dx, al
7318 <1> ;inc bh
7319 <1> ; if(lines==8)
7320 <1> ;cmp bh, 8
7321 00003308 80FF07 <1> cmp bh, 7
7322 0000330B 7506 <1> jne short ssl_1
7323 <1> ; biosfn_set_cursor_shape(0x06,0x07);
7324 0000330D 66B90706 <1> mov cx, 0607h
7325 00003311 EB06 <1> jmp short ssl_2
7326 <1> ssl_1:
7327 <1> ; biosfn_set_cursor_shape(lines-4,lines-3);
7328 00003313 88F9 <1> mov cl, bh ; lines - 1
7329 00003315 88CD <1> mov ch, cl ; lines - 1 (16 -> 15)
7330 00003317 FECD <1> dec ch ; lines - 2 (16 -> 14)
7331 <1> ssl_2:
7332 <1> ; CH = start line, CL = stop line
7333 00003319 B40A <1> mov ah, 10 ; 6845 register for cursor set
7334 0000331B 66890D[37680000] <1> mov [CURSOR_MODE], cx ; save in data area
7335 00003322 E877F0FFFF <1> call m16 ; output cx register
7336 <1> ; write_word(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, lines);
7337 00003327 FEC7 <1> inc bh ; lines
7338 00003329 883D[22680000] <1> mov [CHAR_HEIGHT], bh
7339 <1> ; outb(crtc_addr, 0x12);
7340 0000332F 66BAD403 <1> mov dx, 3D4h ; CRTC_ADDRESS
7341 00003333 B012 <1> mov al, 12h
7342 00003335 EE <1> out dx, al
7343 <1> ; vde = inb(crtc_addr+1);
7344 <1> ;inc dx
7345 <1> ; 02/08/2022
7346 00003336 FEC2 <1> inc dl
7347 00003338 EC <1> in al, dx
7348 00003339 88C4 <1> mov ah, al
7349 <1> ; outb(crtc_addr, 0x07);
7350 <1> ;dec dx
7351 <1> ; 02/08/2022
7352 0000333B FECA <1> dec dl
7353 0000333D B007 <1> mov al, 07h
7354 0000333F EE <1> out dx, al
7355 <1> ; ov1 = inb(crtc_addr+1);
7356 <1> ;inc dx
7357 <1> ; 02/08/2022
7358 00003340 FEC2 <1> inc dl
7359 00003342 EC <1> in al, dx
7360 <1> ; vde += (((ov1 & 0x02) << 7) + ((ov1 & 0x40) << 3) + 1);
7361 00003343 88E2 <1> mov dl, ah ; vde
7362 00003345 88C6 <1> mov dh, al ; ov1
7363 <1> ;and ax, 02h
7364 <1> ;shl ax, 7
7365 <1> ; 02/08/2022

```

```

7366 00003347 31C0      <1>      xor     eax, eax
7367 00003349 88F0      <1>      mov     al, dh
7368 0000334B 2402      <1>      and     al, 2
7369 0000334D C1E007    <1>      shl     eax, 7
7370      <1>      ;mov    cx, ax ; (ovl & 0x02) << 7)
7371      <1>      ; 02/08/2022
7372 00003350 89C1      <1>      mov     ecx, eax
7373 00003352 28E4      <1>      sub     ah, ah
7374 00003354 88F0      <1>      mov     al, dh ; ovl
7375      <1>      ;and    ax, 40h
7376      <1>      ;shl    ax, 3 ; (ovl & 0x40) << 3)
7377      <1>      ;inc    ax ; + 1
7378      <1>      ;add    ax, cx
7379      <1>      ; 02/08/2022
7380 00003356 2440      <1>      and     al, 40h
7381 00003358 C1E003    <1>      shl     eax, 3
7382 0000335B 40        <1>      inc     eax
7383 0000335C 01C8      <1>      add     eax, ecx
7384 0000335E 30F6      <1>      xor     dh, dh
7385      <1>      ;add    ax, dx ; + vde
7386      <1>      ; 02/08/2022
7387 00003360 01D0      <1>      add     eax, edx
7388      <1>      ; rows = vde / lines;
7389 00003362 F6F7      <1>      div     bh
7390      <1>      ;dec    al ; rows -1
7391      <1>      ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, rows-1);
7392 00003364 A2[26680000] <1>      mov     [vga_rows], al ; rows (not 'rows-1' !);
7393      <1>      ; write_word(BIOSMEM_SEG, BIOSMEM_PAGE_SIZE, rows * cols * 2);
7394      <1>      ;mov    ah, [CRT_COLS]
7395      <1>      ;mul    ah
7396      <1>      ; 17/11/2020
7397 00003369 F625[20680000] <1>      mul     byte [CRT_COLS]
7398      <1>      ;shl    ax, 1
7399      <1>      ; 02/08/2022
7400 0000336F D1E0      <1>      shl     eax, 1
7401 00003371 66A3[38830100] <1>      mov     [CRT_LEN], ax
7402 00003377 C3        <1>      retn
7403      <1>
7404      <1>      load_text_8_14_pat:
7405      <1>      ; 02/08/2022 (TRDOS 3386 v2.0.5)
7406      <1>      ; 26/07/2016
7407      <1>      ; 25/07/2016
7408      <1>      ; 23/07/2016
7409      <1>      ; 09/07/2016
7410      <1>      ; load user defined (EGA/VGA) text fonts
7411      <1>      ;
7412      <1>      ; derived from 'Plex86/Bochs VGABios' source code
7413      <1>      ; vgabios-0.7a (2011)
7414      <1>      ; by the LGPL VGABios developers Team (2001-2008)
7415      <1>      ; 'vgabios.c', 'biosfn_load_text_8_14_pat'
7416      <1>
7417      <1>      ; biosfn_load_text_8_14_pat (AL,BL)
7418      <1>
7419      <1>      ; get_font_access();
7420      <1>      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
7421      <1>      ; for(i=0; i<0x100; i++)
7422      <1>      ; {
7423      <1>      ;     src = i * 14;
7424      <1>      ;     dest = blockaddr + i * 32;
7425      <1>      ;     memcpyb(0xA000, dest, 0xC000, vgafont14+src, 14);
7426      <1>      ; }
7427      <1>      ; release_font_access();
7428      <1>      ; if(AL>=0x10)
7429      <1>      ; {
7430      <1>      ;     set_scan_lines(14);
7431      <1>      ; }
7432      <1>
7433 00003378 50        <1>      push    eax
7434 00003379 E80BFFFFFF <1>      call    get_font_access
7435      <1>
7436      <1>      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
7437      <1>      ;mov    dl, bl
7438      <1>      ;and    dl, 3
7439      <1>      ;shl    dx, 14
7440      <1>      ;xchg   dx, bx
7441      <1>      ;and    dl, 4
7442      <1>      ;shl    dx, 11
7443      <1>      ;add    dx, bx
7444      <1>
7445      <1>      ;xor    dx, dx ; blockaddr = 0
7446      <1>      ; Always block 0 for TRDOS 386 ! (blockaddr=0)
7447      <1>
7448 0000337E 28DB      <1>      sub     bl, bl ; i = 0
7449 00003380 B70E      <1>      mov     bh, 14
7450 00003382 BE[30550100] <1>      mov     esi, vgafont14
7451 00003387 BF00000A00 <1>      mov     edi, 0A0000h
7452      <1>      ; 02/08/2022
7453 0000338C 29C9      <1>      sub     ecx, ecx
7454      <1>      lt8_14_1:
7455      <1>      ;mov    al, bl
7456      <1>      ;mul    bh
7457      <1>      ;movzx  esi, ax
7458      <1>      ;add    esi, vgafont14
7459      <1>      ;mov    al, bl
7460      <1>      ;sub    ah, ah
7461      <1>      ;shl    ax, 5 ; * 32
7462      <1>      ;add    ax, dx ; blockaddr + i * 32;
7463      <1>      ;movzx  edi, ax ; dest
7464      <1>      ;add    edi, 0A0000h
7465      <1>      ; 02/08/2022
7466      <1>      ;movzx  ecx, bh
7467 0000338E 88F9      <1>      mov     cl, bh
7468 00003390 F3A4      <1>      rep     movsb
7469 00003392 83C712    <1>      add     edi, 18 ; 32 - 14
7470 00003395 FEC3      <1>      inc     bl
7471 00003397 75F5      <1>      jnz     short lt8_14_1
7472      <1>      ;
7473 00003399 E81EFFFFFF <1>      call    release_font_access
7474      <1>      ;
7475 0000339E 58        <1>      pop     eax
7476      <1>      ; if(AL>=0x10)
7477 0000339F 3C10      <1>      cmp     al, 10h
7478 000033A1 7205      <1>      jb     short lt8_14_4
7479      <1>      ; BH = 14
7480      <1>      ; set_scan_lines(14);
7481 000033A3 E84FFFFFFF <1>      call    set_scan_lines
7482      <1>      lt8_14_4:
7483 000033A8 C3        <1>      retn
7484      <1>
7485      <1>      load_text_8_8_pat:
7486      <1>      ; 02/08/2022 (TRDOS 3386 v2.0.5)
7487      <1>      ; 05/01/2021 (TRDOS 386 v2.0.3)
7488      <1>      ; 26/07/2016
7489      <1>      ; 25/07/2016

```

```

7490      <1>      ; 23/07/2016
7491      <1>      ; 09/07/2016
7492      <1>      ; load user defined (EGA/VGA) text fonts
7493      <1>      ;
7494      <1>      ; derived from 'Plex86/Bochs VGABios' source code
7495      <1>      ; vgabios-0.7a (2011)
7496      <1>      ; by the LGPL VGABios developers Team (2001-2008)
7497      <1>      ; 'vgabios.c', 'biosfn_load_text_8_8_pat'
7498      <1>
7499      <1>      ; biosfn_load_text_8_8_pat (AL,BL)
7500      <1>
7501      <1>      ; get_font_access();
7502      <1>      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
7503      <1>      ; for(i=0;i<0x100;i++)
7504      <1>      ; {
7505      <1>      ;     src = i * 8;
7506      <1>      ;     dest = blockaddr + i * 32;
7507      <1>      ;     memcpyb(0xA000, dest, 0xC000, vgafont8+src, 8);
7508      <1>      ; }
7509      <1>      ; release_font_access();
7510      <1>      ; if(AL>=0x10)
7511      <1>      ; {
7512      <1>      ;     set_scan_lines(8);
7513      <1>      ; }
7514      <1>
7515      000033A9 50      <1>      push     eax
7516      000033AA E8DAFEFFFF <1>      call    get_font_access
7517      <1>
7518      <1>      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
7519      <1>      ; mov     dl, bl
7520      <1>      ; and     dl, 3
7521      <1>      ; shl     dx, 14
7522      <1>      ; xchg    dx, bx
7523      <1>      ; and     dl, 4
7524      <1>      ; shl     dx, 11
7525      <1>      ; add     dx, bx
7526      <1>
7527      <1>      ; xor     dx, dx ; blockaddr = 0
7528      <1>      ; Always block 0 for TRDOS 386 ! (blockaddr=0)
7529      <1>
7530      000033AF 28DB      <1>      sub     bl, bl ; i = 0
7531      000033B1 B708      <1>      mov     bh, 8
7532      <1>      ; mov     esi, vgafont8
7533      000033B3 BF0000A00 <1>      mov     edi, 0A0000h
7534      <1>
7535      <1>      ; 02/08/2022
7536      000033B8 29C9      <1>      sub     ecx, ecx
7537      <1>
7538      <1>      ; 05/01/2021
7539      000033BA F605[6E9D0100]80 <1>      test    byte [ufont], 80h
7540      000033C1 7410      <1>      jz      short lt8_8_0
7541      <1>      ; user font permission (after set mode)
7542      000033C3 F605[6E9D0100]08 <1>      test    byte [ufont], 8
7543      000033CA 7407      <1>      jz      short lt8_8_0
7544      000033CC BE00500900 <1>      mov     esi, VGAFONT8USER
7545      000033D1 EB05      <1>      jmp     short lt8_8_1
7546      <1>      lt8_8_0:
7547      000033D3 BE[304D0100] <1>      mov     esi, vgafont8
7548      <1>      lt8_8_1:
7549      <1>      ; mov     al, bl
7550      <1>      ; mul     bh
7551      <1>      ; movzx   esi, ax
7552      <1>      ; add     esi, vgafont8
7553      <1>      ; mov     al, bl
7554      <1>      ; sub     ah, ah
7555      <1>      ; shl     ax, 5 ; * 32
7556      <1>      ; add     ax, dx ; blockaddr + i * 32;
7557      <1>      ; movzx   edi, ax ; dest
7558      <1>      ; add     edi, 0A0000h
7559      <1>      ; 02/08/2022
7560      <1>      ; movzx   ecx, bh
7561      000033D8 88F9      <1>      mov     cl, bh
7562      000033DA F3A4      <1>      rep     movsb
7563      000033DC 83C718      <1>      add     edi, 24 ; 32 - 8
7564      000033DF FEC3      <1>      inc     bl
7565      000033E1 75F5      <1>      jnz     short lt8_8_1
7566      <1>
7567      000033E3 E8D4FEFFFF <1>      call    release_font_access
7568      <1>
7569      000033E8 58      <1>      pop     eax
7570      <1>      ; if(AL>=0x10)
7571      000033E9 3C10      <1>      cmp     al, 10h
7572      000033EB 7205      <1>      jb      short lt8_8_2
7573      <1>      ; BH = 8
7574      <1>      ; set_scan_lines(8);
7575      000033ED E805FFFFFF <1>      call    set_scan_lines
7576      <1>      lt8_8_2:
7577      000033F2 C3      <1>      retn
7578      <1>
7579      <1>      load_text_8_16_pat:
7580      <1>      ; 02/08/2022 (TRDOS 386 v2.0.5)
7581      <1>      ; 05/01/2021 (TRDOS 386 v2.0.3)
7582      <1>      ; 26/07/2016
7583      <1>      ; 25/07/2016
7584      <1>      ; 23/07/2016
7585      <1>      ; 09/07/2016
7586      <1>      ; load user defined (EGA/VGA) text fonts
7587      <1>      ;
7588      <1>      ; derived from 'Plex86/Bochs VGABios' source code
7589      <1>      ; vgabios-0.7a (2011)
7590      <1>      ; by the LGPL VGABios developers Team (2001-2008)
7591      <1>      ; 'vgabios.c', 'biosfn_load_text_8_16_pat'
7592      <1>
7593      <1>      ; biosfn_load_text_8_16_pat (AL,BL)
7594      <1>
7595      <1>      ; get_font_access();
7596      <1>      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
7597      <1>      ; for(i=0;i<0x100;i++)
7598      <1>      ; {
7599      <1>      ;     src = i * 16;
7600      <1>      ;     dest = blockaddr + i * 32;
7601      <1>      ;     memcpyb(0xA000, dest, 0xC000, vgafont16+src, 16);
7602      <1>      ; }
7603      <1>      ; release_font_access();
7604      <1>      ; if(AL>=0x10)
7605      <1>      ; {
7606      <1>      ;     set_scan_lines(16);
7607      <1>      ; }
7608      <1>
7609      000033F3 50      <1>      push     eax
7610      000033F4 E890FEFFFF <1>      call    get_font_access
7611      <1>
7612      <1>      ; blockaddr = ((BL & 0x03) << 14) + ((BL & 0x04) << 11);
7613      <1>      ; mov     dl, bl

```

```

7614      <1>      ;and    dl, 3
7615      <1>      ;shl    dx, 14
7616      <1>      ;xchg   dx, bx
7617      <1>      ;and    dl, 4
7618      <1>      ;shl    dx, 11
7619      <1>      ;add    dx, bx
7620      <1>
7621      <1>      ;xor    dx, dx ; blockaddr = 0
7622      <1>      ; Always block 0 for TRDOS 386 ! (blockaddr=0)
7623      <1>
7624      000033F9 28DB      <1>      sub    b1, b1 ; i = 0
7625      000033FB B710      <1>      mov    bh, 16
7626      <1>      ;mov    esi, vgafont16
7627      000033FD BF00000A00 <1>      mov    edi, 0A0000h
7628      <1>      ;movzx  eax, bh
7629      <1>      ; 02/08/2022
7630      00003402 29C0      <1>      sub    eax, eax
7631      00003404 88F8      <1>      mov    al, bh
7632      <1>
7633      <1>      ; 05/01/2021
7634      00003406 F605[6E9D0100]80 <1>      test   byte [ufont], 80h
7635      0000340D 7410      <1>      jz     short lt8_16_0
7636      <1>      ; user font permission (after set mode)
7637      0000340F F605[6E9D0100]10 <1>      test   byte [ufont], 16
7638      00003416 7407      <1>      jz     short lt8_16_0
7639      00003418 BE00400900 <1>      mov    esi, VGA_FONT16USER
7640      0000341D EB05      <1>      jmp    short lt8_16_1
7641      <1>      lt8_16_0:
7642      0000341F BE[30630100] <1>      mov    esi, vgafont16
7643      <1>      lt8_16_1:
7644      <1>      ;mov    al, b1
7645      <1>      ;mul    bh
7646      <1>      ;movzx  esi, ax
7647      <1>      ;add    esi, vgafont16
7648      <1>      ;mov    al, b1 ; i
7649      <1>      ;sub    ah, ah
7650      <1>      ;shl    ax, 5 ; * 32
7651      <1>      ;add    ax, dx ; blockaddr + i * 32;
7652      <1>      ;movzx  edi, ax ; dest
7653      <1>      ;add    edi, 0A0000h
7654      <1>      ;movzx  ecx, bh
7655      00003424 89C1      <1>      mov    ecx, eax ; 16
7656      00003426 F3A4      <1>      rep    movsb
7657      00003428 01C7      <1>      add    edi, eax ; add edi, 16
7658      0000342A FEC3      <1>      inc    b1
7659      0000342C 75F6      <1>      jnz    short lt8_16_1
7660      <1>
7661      0000342E E889FEFFFF <1>      call   release_font_access
7662      <1>
7663      00003433 58      <1>      pop    eax
7664      <1>      ; if(AL>=0x10)
7665      00003434 3C10      <1>      cmp    al, 10h
7666      00003436 7205      <1>      jb     short lt8_16_2
7667      <1>      ; BH = 16
7668      <1>      ; set_scan_lines(16);
7669      00003438 E8BAFEFFFF <1>      call   set_scan_lines
7670      <1>      lt8_16_2:
7671      0000343D C3      <1>      retn
7672      <1>
7673      <1>      load_gfx_user_chars:
7674      <1>      ; 08/01/2021
7675      <1>      ; 05/01/2021 (TRDOS 386 v2.0.3)
7676      <1>      ; 08/08/2016
7677      <1>      ; 10/07/2016
7678      <1>      ; Setup User-Defined Font for Graphics Mode (VGA)
7679      <1>      ;
7680      <1>      ; derived from 'plex86/Bochs VGABios' source code
7681      <1>      ; vgabios-0.7a (2011)
7682      <1>      ; by the LGPL VGABios developers Team (2001-2008)
7683      <1>      ; 'vgabios.c', 'biosfn_load_gfx_user_chars'
7684      <1>
7685      <1>      ; biosfn_load_gfx_user_chars (ES,BP,CX,BL,DL)
7686      <1>      ; /* set 0x43 INT pointer */
7687      <1>      ; write_word(0x0, 0x43*4, BP);
7688      <1>      ; write_word(0x0, 0x43*4+2, ES);
7689      <1>
7690      <1>      ; 08/01/2021
7691      <1>
7692      <1>      ; BL screen rows code: 00H = user-specified (in DL)
7693      <1>      ;
7694      <1>      ;
7695      <1>      ;
7696      <1>      ; CX bytes per character definition
7697      <1>      ; DL (when BL=0) custom number of character rows on screen
7698      <1>      ; EBP address of font-definition information (user's mem space)
7699      <1>
7700      <1>      ; 05/01/2021
7701      <1>      ;xor    eax, eax
7702      <1>      ;dec    eax ; 0FFFFFFFh (user defined fonts)
7703      <1>      ;mov    [VGA_INT43H], eax
7704      <1>
7705      <1>      ; 08/01/2021
7706      <1>      ; ebp = video font data (buffer) address
7707      0000343E 892D[4A830100] <1>      mov    [VGA_INT43H], ebp
7708      <1>
7709      <1>      ; switch (BL) {
7710      <1>      ; case 0:
7711      <1>      ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, DL-1);
7712      <1>      ; break;
7713      00003444 20DB      <1>      and    b1, b1
7714      00003446 7508      <1>      jnz    short l_gfx_uc_1
7715      00003448 8815[26680000] <1>      mov    [VGA_ROWS], dl ; not DL-1 !
7716      0000344E EB23      <1>      jmp    short l_gfx_uc_4
7717      <1>      l_gfx_uc_1:
7718      <1>      ; case 1:
7719      <1>      ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 13);
7720      <1>      ; break;
7721      00003450 FECB      <1>      dec    b1
7722      00003452 7509      <1>      jnz    short l_gfx_uc_2
7723      <1>      ; b1 = 1
7724      00003454 C605[26680000]0E <1>      mov    byte [VGA_ROWS], 14 ; not 13 !
7725      0000345B EB16      <1>      jmp    short l_gfx_uc_4
7726      <1>      l_gfx_uc_2:
7727      0000345D FECB      <1>      dec    b1
7728      0000345F 740B      <1>      jz     short l_gfx_uc_3 ; b1 = 2
7729      00003461 FECB      <1>      dec    b1
7730      00003463 750E      <1>      jnz    short l_gfx_uc_4 ; b1 > 3
7731      <1>      ; b1 = 3
7732      <1>      ; case 3:
7733      <1>      ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 42);
7734      <1>      ; break;
7735      00003465 C605[26680000]2B <1>      mov    byte [VGA_ROWS], 43 ; not 42 !
7736      <1>      l_gfx_uc_3:
7737      <1>      ; case 2:

```



```

7738      <1>      ; default:
7739      <1>      ;     write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 24);
7740      <1>      ;     break;
7741      <1>      ;     b1 = 2 or b1 > 3
7742 0000346C C605[26680000]19 <1>      mov     byte [VGA_ROWS], 25 ; not 24 !
7743      <1>      ; }
7744      <1>      l_gfx_uc_4:
7745      <1>      ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, CX);
7746 00003473 880D[22680000] <1>      mov     [CHAR_HEIGHT], cl
7747      <1>      ; }
7748 00003479 C3 <1>      retn
7749      <1>
7750      <1>      load_gfx_8_14_chars:
7751      <1>      ; 08/08/2016
7752      <1>      ; 10/07/2016
7753      <1>      ; Setup ROM 8x14 Font for Graphics Mode (VGA)
7754      <1>      ;
7755      <1>      ; derived from 'Plex86/Bochs VGABios' source code
7756      <1>      ; vgabios-0.7a (2011)
7757      <1>      ; by the LGPL VGABios developers Team (2001-2008)
7758      <1>      ; 'vgabios.c', 'biosfn_load_gfx_8_14_chars'
7759      <1>
7760      <1>      ; biosfn_load_gfx_8_14_chars (BL)
7761      <1>      ; /* set 0x43 INT pointer */
7762      <1>      ; write_word(0x0, 0x43*4, &vgafont14);
7763      <1>      ; write_word(0x0, 0x43*4+2, 0xC000);
7764 0000347A C705[4A830100]- <1>      mov     dword [VGA_INT43H], vgafont14
7765 00003480 [30550100] <1>
7766      <1>
7767      <1>      ; BL      screen rows code: 00H = user-specified (in DL)
7768      <1>      ;          01H = 14 rows
7769      <1>      ;          02H = 25 rows
7770      <1>      ;          03H = 43 rows
7771      <1>      ; DL      (when BL=0) custom number of char rows on screen
7772      <1>
7773      <1>      ; switch (BL) {
7774      <1>      ; case 0:
7775      <1>      ;     write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, DL-1);
7776      <1>      ;     break;
7777 00003484 20DB <1>      and     b1, b1
7778 00003486 7508 <1>      jnz     short l_gfx_8_14c_1
7779 00003488 8815[26680000] <1>      mov     [VGA_ROWS], dl ; not DL-1 !
7780 0000348E EB23 <1>      jmp     short l_gfx_8_14c_4
7781      <1>      l_gfx_8_14c_1:
7782      <1>      ; case 1:
7783      <1>      ;     write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 13);
7784      <1>      ;     break;
7785 00003490 FECB <1>      dec     b1
7786 00003492 7509 <1>      jnz     short l_gfx_8_14c_2
7787      <1>      ; b1 = 1
7788 00003494 C605[26680000]0E <1>      mov     byte [VGA_ROWS], 14 ; not 13 !
7789 0000349B EB16 <1>      jmp     short l_gfx_8_14c_4
7790      <1>      l_gfx_8_14c_2:
7791      <1>      dec     b1
7792 0000349D FECB <1>      jz      short l_gfx_8_14c_3 ; b1 = 2
7793 0000349F 740B <1>      dec     b1
7794 000034A1 FECB <1>      jnz     short l_gfx_8_14c_4 ; b1 > 3
7795 000034A3 750E <1>      ; b1 = 3
7796      <1>      ; case 3:
7797      <1>      ;     write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 42);
7798      <1>      ;     break;
7799 000034A5 C605[26680000]2B <1>      mov     byte [VGA_ROWS], 43 ; not 42 !
7800      <1>      l_gfx_8_14c_3:
7801      <1>      ; case 2:
7802      <1>      ; default:
7803      <1>      ;     write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 24);
7804      <1>      ;     break;
7805      <1>      ; b1 = 2 or b1 > 3
7806 000034AC C605[26680000]19 <1>      mov     byte [VGA_ROWS], 25 ; not 24 !
7807      <1>      ; }
7808      <1>      l_gfx_8_14c_4:
7809      <1>      ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 14);
7810      <1>      mov     byte [CHAR_HEIGHT], 14
7811      <1>      ; }
7812      <1>      retn
7813      <1>
7814      <1>      load_gfx_8_8_chars:
7815      <1>      ; 08/08/2016
7816      <1>      ; 10/07/2016
7817      <1>      ; Setup ROM 8x14 Font for Graphics Mode (VGA)
7818      <1>      ;
7819      <1>      ; derived from 'Plex86/Bochs VGABios' source code
7820      <1>      ; vgabios-0.7a (2011)
7821      <1>      ; by the LGPL VGABios developers Team (2001-2008)
7822      <1>      ; 'vgabios.c', 'biosfn_load_gfx_8_8_dd_chars'
7823      <1>
7824      <1>      ; biosfn_load_gfx_8_8_dd_chars (BL)
7825      <1>      ; /* set 0x43 INT pointer */
7826      <1>      ; write_word(0x0, 0x43*4, &vgafont8);
7827      <1>      ; write_word(0x0, 0x43*4+2, 0xC000);
7828 000034BB C705[4A830100]- <1>      mov     dword [VGA_INT43H], vgafont8
7829 000034C1 [304D0100] <1>
7830      <1>
7831      <1>      ; BL      screen rows code: 00H = user-specified (in DL)
7832      <1>      ;          01H = 14 rows
7833      <1>      ;          02H = 25 rows
7834      <1>      ;          03H = 43 rows
7835      <1>      ; DL      (when BL=0) custom number of char rows on screen
7836      <1>
7837      <1>      ; switch (BL) {
7838      <1>      ; case 0:
7839      <1>      ;     write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, DL-1);
7840      <1>      ;     break;
7841      <1>      ;     and     b1, b1
7842 000034C5 20DB <1>      jnz     short l_gfx_8_8c_1
7843 000034C7 7508 <1>      mov     [VGA_ROWS], dl ; not DL-1 !
7844 000034C9 8815[26680000] <1>      jmp     short l_gfx_8_8c_4
7845 000034CF EB23 <1>      jmp     short l_gfx_8_8c_1
7846      <1>      l_gfx_8_8c_1:
7847      <1>      ; case 1:
7848      <1>      ;     write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 13);
7849      <1>      ;     break;
7850      <1>      dec     b1
7851 000034D1 FECB <1>      jnz     short l_gfx_8_8c_2
7852 000034D3 7509 <1>      ; b1 = 1
7853 000034D5 C605[26680000]0E <1>      mov     byte [VGA_ROWS], 14 ; not 13 !
7854 000034DC EB16 <1>      jmp     short l_gfx_8_8c_4
7855      <1>      l_gfx_8_8c_2:
7856      <1>      dec     b1
7857      <1>      jz      short l_gfx_8_8c_3 ; b1 = 2
7858      <1>      dec     b1
7859      <1>      jnz     short l_gfx_8_8c_4 ; b1 > 3

```

```

7860             ; break;
7861 000034E6 C605[26680000]2B <1> mov byte [VGA_ROWS], 43 ; not 42 !
7862 <1> l_gfx_8_8c_3:
7863 <1> ; case 2:
7864 <1> ; default:
7865 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 24);
7866 <1> ; break;
7867 <1> ; b1 = 2 or b1 > 3
7868 000034ED C605[26680000]19 <1> mov byte [VGA_ROWS], 25 ; not 24 !
7869 <1> ; }
7870 <1> l_gfx_8_8c_4:
7871 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 8);
7872 000034F4 C605[22680000]08 <1> mov byte [CHAR_HEIGHT], 8
7873 <1> ; }
7874 000034FB C3 <1> retn
7875 <1>
7876 <1> load_gfx_8_16_chars:
7877 <1> ; 08/08/2016
7878 <1> ; 10/07/2016
7879 <1> ; Setup ROM 8x14 Font for Graphics Mode (VGA)
7880 <1> ;
7881 <1> ; derived from 'Plex86/Bochs VGABios' source code
7882 <1> ; vgabios-0.7a (2011)
7883 <1> ; by the LGPL VGABios developers Team (2001-2008)
7884 <1> ; 'vgabios.c', 'biosfn_load_gfx_8_16_chars'
7885 <1>
7886 <1> ; biosfn_load_gfx_8_16_chars (BL)
7887 <1> ; /* set 0x43 INT pointer */
7888 <1> ; write_word(0x0, 0x43*4, &vgafont16);
7889 <1> ; write_word(0x0, 0x43*4+2, 0xC000);
7890 000034FC C705[4A830100]- <1> mov dword [VGA_INT43H], vgafont16
7891 00003502 [30630100] <1>
7892 <1>
7893 <1> ; BL screen rows code: 00H = user-specified (in DL)
7894 <1> ; ; 01H = 14 rows
7895 <1> ; ; 02H = 25 rows
7896 <1> ; ; 03H = 43 rows
7897 <1> ; DL (when BL=0) custom number of char rows on screen
7898 <1>
7899 <1> ; switch (BL) {
7900 <1> ; case 0:
7901 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, DL-1);
7902 <1> ; break;
7903 <1> and b1, b1
7904 00003506 20DB <1> jnz short l_gfx_8_16c_1
7905 0000350A 8815[26680000] <1> mov [VGA_ROWS], dl ; not DL-1 !
7906 00003510 EB23 <1> jmp short l_gfx_8_16c_4
7907 <1> l_gfx_8_16c_1:
7908 <1> ; case 1:
7909 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 13);
7910 <1> ; break;
7911 00003512 FECB <1> dec b1
7912 00003514 7509 <1> jnz short l_gfx_8_16c_2
7913 <1> ; b1 = 1
7914 00003516 C605[26680000]0E <1> mov byte [VGA_ROWS], 14 ; not 13 !
7915 0000351D EB16 <1> jmp short l_gfx_8_16c_4
7916 <1> l_gfx_8_16c_2:
7917 0000351F FECB <1> dec b1
7918 00003521 740B <1> jz short l_gfx_8_16c_3 ; b1 = 2
7919 00003523 FECB <1> dec b1
7920 00003525 750E <1> jnz short l_gfx_8_16c_4 ; b1 > 3
7921 <1> ; b1 = 3
7922 <1> ; case 3:
7923 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 42);
7924 <1> ; break;
7925 00003527 C605[26680000]2B <1> mov byte [VGA_ROWS], 43 ; not 42 !
7926 <1> l_gfx_8_16c_3:
7927 <1> ; case 2:
7928 <1> ; default:
7929 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS, 24);
7930 <1> ; break;
7931 <1> ; b1 = 2 or b1 > 3
7932 0000352E C605[26680000]19 <1> mov byte [VGA_ROWS], 25 ; not 24 !
7933 <1> ; }
7934 <1> l_gfx_8_16c_4:
7935 00003535 C605[22680000]10 <1> ; write_byte(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT, 16);
7936 <1> mov byte [CHAR_HEIGHT], 16
7937 0000353C C3 <1> ; }
7938 <1> retn
7939 <1>
7940 <1> get_font_info:
7941 <1> ; 03/08/2022 (TRDOS 386 v2.0.5)
7942 <1> ; 08/01/2021 (TRDOS 386 v2.0.3)
7943 <1> ; 19/09/2016
7944 <1> ; 08/08/2016
7945 <1> ; 10/07/2016
7946 <1> ; Get Current Character Generator Info (VGA)
7947 <1> ;
7948 <1> ; derived from 'Plex86/Bochs VGABios' source code
7949 <1> ; vgabios-0.7a (2011)
7950 <1> ; by the LGPL VGABios developers Team (2001-2008)
7951 <1> ; 'vgabios.c', 'biosfn_get_font_info'
7952 <1>
7953 <1> ; Modified for TRDOS 386 !
7954 <1>
7955 <1> INPUT ->
7956 <1> AX = 1130h
7957 <1> BL = 0 -> Get info for current VGA font
7958 <1> ; (BH = unused)
7959 <1> 19/09/2016
7960 <1> BL > 0 -> Get requested character font data
7961 <1> BL = 1 -> vgafont8
7962 <1> BL = 2 -> vgafont14
7963 <1> BL = 3 -> vgafont16
7964 <1> ; 08/01/2021
7965 <1> BL = 4 -> user defined 8x8 font
7966 <1> BL = 5 -> user defined 8x14 font
7967 <1> BL = 6 -> user defined 8x16 font
7968 <1> BL > 6 -> Invalid function (for now!)
7969 <1> BH = ASCII code of the first character
7970 <1> ECX = Number of characters from the 1st char
7971 <1> ECX >= 256 -> All (256-BH) characters
7972 <1> ECX = 0 -> All characters (BH = unused)
7973 <1> EDX = User's Buffer Address
7974 <1> OUTPUT ->
7975 <1> AL = height (scanlines), bytes per character
7976 <1> AH = screen rows
7977 <1> Byte 16-23 of EAX = number of columns
7978 <1> Byte 24-31 of EAX =
7979 <1> 0 -> default font (not configured yet)
7980 <1> 0FFh -> user defined font
7981 <1> 14 = vgafont14
7982 <1> 8 = vgafont8
7983 <1> 16 = vgafont16

```

```

7983      <1>      ; If BL input > 0 ->
7984      <1>      ; EAX = Actual transfer count
7985      <1>      ;
7986      <1>      and     bl, bl
7987      <1>      jz      short gfi_1
7988      <1>      ; invalid function (input)
7989      <1>      ; 08/01/2021
7990      <1>      cmp     bl, 4
7991      <1>      jb      short gfi_5
7992      <1>      je      short gfi_3
7993      <1>      cmp     bl, 6
7994      <1>      je      short gfi_4
7995      <1>      ; bh = 5 or bh > 6
7996      <1>      gfi_0:
7997      <1>      xor     eax, eax ; 0
7998      <1>      retn
7999      <1>      gfi_1:
8000      <1>      mov     al, [CHAR_HEIGHT]
8001      <1>      mov     ah, [VGA_ROWS]
8002      <1>      shl     eax, 16
8003      <1>      mov     al, [CRT_COLS]
8004      <1>      mov     ecx, [VGA_INT43H]
8005      <1>      and     ecx, ecx
8006      <1>      jz      short gfi_2 ; 0 = default font
8007      <1>      ; 08/01/2021
8008      <1>      dec     ah ; 0FFh
8009      <1>      cmp     ecx, VGA_FONT16USER
8010      <1>      je      short gfi_2
8011      <1>      cmp     ecx, VGA_FONT8USER
8012      <1>      je      short gfi_2
8013      <1>      mov     ah, [CHAR_HEIGHT] ; font size = height
8014      <1>      gfi_2:
8015      <1>      rol     eax, 16
8016      <1>      retn
8017      <1>      gfi_3:
8018      <1>      ; 08/01/2021
8019      <1>      test    byte [ufont], 08h ; 8x8 user font
8020      <1>      jz      short gfi_0 ; not loaded !
8021      <1>      mov     esi, VGA_FONT8USER ; *
8022      <1>      ;mov     bl, 8
8023      <1>      ;jmp     short gfi_8
8024      <1>      jmp     short gfi_10
8025      <1>      gfi_4:
8026      <1>      ; 08/01/2021
8027      <1>      test    byte [ufont], 10h ; 8x16 user font
8028      <1>      jz      short gfi_0 ; not loaded !
8029      <1>      mov     esi, VGA_FONT16USER ; *
8030      <1>      jmp     short gfi_7
8031      <1>      gfi_5:
8032      <1>      cmp     bl, 2
8033      <1>      jb      short gfi_9
8034      <1>      ja      short gfi_6
8035      <1>      ; BL = 2 -> vgaFont14
8036      <1>      mov     esi, vgaFont14 ; *
8037      <1>      mov     bl, 14
8038      <1>      jmp     short gfi_8
8039      <1>      gfi_6:
8040      <1>      ; BL = 3 -> vgaFont16
8041      <1>      mov     esi, vgaFont16 ; *
8042      <1>      gfi_7:
8043      <1>      mov     bl, 16
8044      <1>      gfi_8:
8045      <1>      mov     edi, edx ; **
8046      <1>      or      ecx, ecx
8047      <1>      jz      short gfi_11 ; all chars from the 00h
8048      <1>      ;mov     al, bh ; character index
8049      <1>      ; 03/08/2022
8050      <1>      movzx   eax, bh
8051      <1>      mul     bl ; char index * char height/size
8052      <1>      ;movzx   edx, ax
8053      <1>      ; 03/08/2022
8054      <1>      add     esi, edx ; *
8055      <1>      add     esi, eax
8056      <1>      xor     edx, edx
8057      <1>      dec     dl
8058      <1>      ; edx = 0FFh = 255
8059      <1>      ;mov     dx, 255
8060      <1>      sub     dl, bh
8061      <1>      ;inc     dx
8062      <1>      ; 03/08/2022
8063      <1>      inc     edx
8064      <1>      cmp     ecx, edx
8065      <1>      ja      short gfi_11
8066      <1>      je      short gfi_12
8067      <1>      mov     ecx, edx
8068      <1>      jmp     short gfi_12
8069      <1>      gfi_9:
8070      <1>      ;BL = 1 -> vgaFont8
8071      <1>      mov     esi, vgaFont8 ; *
8072      <1>      gfi_10:
8073      <1>      mov     bl, 8
8074      <1>      jmp     short gfi_8
8075      <1>      gfi_11:
8076      <1>      ;mov     ecx, 256
8077      <1>      ; 03/08/2022
8078      <1>      sub     ecx, ecx
8079      <1>      inc     ch
8080      <1>      ; ecx = 100h
8081      <1>      gfi_12:
8082      <1>      ; 08/01/2021
8083      <1>      mov     eax, ecx ; character count
8084      <1>      xor     bh, bh
8085      <1>      mul     bx ; char count * char height/size
8086      <1>      mov     ecx, eax
8087      <1>
8088      <1>      ; ESI = source address in system space
8089      <1>      ; EDI = user's buffer address
8090      <1>      ; ECX = transfer (byte) count
8091      <1>      call    transfer_to_user_buffer
8092      <1>      mov     eax, ecx ; actual transfer count
8093      <1>      retn
8094      <1>
8095      <1>      set_all_palette_reg:
8096      <1>      ; 03/08/2022
8097      <1>      ; 10/08/2016
8098      <1>      ; Set All Palette Registers and Overscan
8099      <1>      ; EDX = Address of 17 bytes;
8100      <1>      ; an rgbRGB value for each of 16 palette
8101      <1>      ; registers plus one for the border.
8102      <1>
8103      <1>      mov     esi, edx ; user buffer
8104      <1>      ;mov     ecx, 17
8105      <1>      ; 03/08/2022
8106      <1>      sub     ecx, ecx

```

```

8107 00003600 B111      <1>      mov     cl, 17
8108 00003602 89E7      <1>      mov     edi, esp
8109 00003604 83EC14    <1>      sub     esp, 20
8110 00003607 E849D50000 <1>      call    transfer_from_user_buffer
8111                                <1>      ;jc     VIDEO_RETURN
8112                                <1>
8113 0000360C 66BADA03 <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
8114 00003610 EC          <1>      in      al, dx
8115                                <1>      ;mov    cl, 0
8116                                <1>      ; 03/08/2022
8117 00003611 28C9      <1>      sub     cl, cl
8118                                <1>      ;mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
8119                                <1>      ; 03/08/2022
8120 00003613 B2C0      <1>      mov     dl, 0C0h
8121                                <1>      set_palette_loop:
8122 00003615 88C8      <1>      mov     al, cl
8123 00003617 EE          <1>      out     dx, al
8124 00003618 8A07      <1>      mov     al, [edi]
8125 0000361A EE          <1>      out     dx, al
8126 0000361B 47          <1>      inc     edi
8127 0000361C FEC1      <1>      inc     cl
8128 0000361E 80F910    <1>      cmp     cl, 10h
8129 00003621 75F2      <1>      jne     short set_palette_loop
8130 00003623 B011      <1>      mov     al, 11h
8131 00003625 EE          <1>      out     dx, al
8132 00003626 8A07      <1>      mov     al, [edi]
8133 00003628 EE          <1>      out     dx, al
8134 00003629 B020      <1>      mov     al, 20h
8135 0000362B EE          <1>      out     dx, al
8136                                <1>      ; ifdef VBOX
8137                                <1>      ;mov    dx, 3DAh ; VGAREG_ACTL_RESET
8138                                <1>      ; 03/08/2022
8139 0000362C B2DA      <1>      mov     dl, 0DAh
8140 0000362E EC          <1>      in      al, dx
8141                                <1>      ; endif ; VBOX
8142 0000362F 83C414    <1>      add     esp, 20
8143 00003632 E941E5FFFF <1>      jmp     VIDEO_RETURN
8144                                <1>
8145                                <1>      ; 07/08/2022
8146                                <1>      toggle_intensity:
8147                                <1>      ; 03/08/2022
8148                                <1>      ; 10/08/2016
8149                                <1>      ; Select Foreground Blink or Bold Background
8150                                <1>      ; BL = 00h = enable bold backgrounds
8151                                <1>      ;           (16 background colors)
8152                                <1>      ;           ;           01h = enable blinking foreground
8153                                <1>      ;           (8 background colors)
8154                                <1>
8155 00003637 66BADA03 <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
8156 0000363B EC          <1>      in      al, dx
8157                                <1>      ;mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
8158                                <1>      ; 03/08/2022
8159 0000363C B2C0      <1>      mov     dl, 0C0h
8160 0000363E B010      <1>      mov     al, 10h
8161 00003640 EE          <1>      out     dx, al
8162                                <1>      ;mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
8163                                <1>      ; 03/08/2022
8164 00003641 FEC2      <1>      inc     dl ; dx = 3C1h
8165 00003643 EC          <1>      in      al, dx
8166 00003644 24F7      <1>      and     al, 0F7h
8167 00003646 80E301    <1>      and     bl, 01h
8168 00003649 C0E303    <1>      shl     bl, 3
8169 0000364C 08D8      <1>      or      al, bl
8170                                <1>      ;mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
8171                                <1>      ; 03/08/2022
8172 0000364E FECA      <1>      dec     dl ; dx = 3C0h
8173 00003650 EE          <1>      out     dx, al
8174 00003651 B020      <1>      mov     al, 20h
8175 00003653 EE          <1>      out     dx, al
8176                                <1>      ; ifdef VBOX
8177                                <1>      ;mov    dx, 3DAh ; VGAREG_ACTL_RESET
8178                                <1>      ; 03/08/2022
8179 00003654 B2DA      <1>      mov     dl, 0DAh
8180 00003656 EC          <1>      in      al, dx
8181                                <1>      ; endif ; VBOX
8182 00003657 E91CE5FFFF <1>      jmp     VIDEO_RETURN
8183                                <1>
8184                                <1>      ; 07/08/2022
8185                                <1>      vga_palf_unknown:
8186 0000365C 29C0      <1>      sub     eax, eax ; 0 = invalid function
8187 0000365E E91AE5FFFF <1>      jmp     _video_return
8188                                <1>
8189                                <1>      ; 07/08/2022
8190                                <1>      vga_palf_101B:
8191 00003663 3C1B      <1>      cmp     al, 1Bh
8192                                <1>      ;jne     short vga_palf_unknown
8193 00003665 77F5      <1>      ja      short vga_palf_unknown
8194                                <1>
8195 00003667 E810F6FFFF <1>      call    gray_scale_summing
8196 0000366C E907E5FFFF <1>      jmp     VIDEO_RETURN
8197                                <1>
8198                                <1>      vga_pal_funcs:
8199                                <1>      ; 07/08/2022
8200                                <1>      ; 10/08/2016
8201                                <1>      ; VGA Palette functions
8202                                <1>      ;
8203                                <1>      ; derived from 'Plex86/Bochs VGABios' source code
8204                                <1>      ; vgabios-0.7a (2011)
8205                                <1>      ; by the LGPL VGABios developers Team (2001-2008)
8206                                <1>      ; 'vgabios.c', 'vgarom.asm'
8207                                <1>
8208 00003671 3C00      <1>      cmp     al, 0
8209 00003673 7467      <1>      je      short set_single_palette_reg
8210                                <1>      vga_palf_1002:
8211 00003675 3C02      <1>      cmp     al, 2
8212 00003677 7483      <1>      je      short set_all_palette_reg
8213                                <1>      ; 07/08/2022
8214 00003679 7702      <1>      ja      short vga_palf_1003
8215 0000367B EB5D      <1>      jmp     short set_overscan_border_color
8216                                <1>      ; 07/08/2022
8217                                <1>      ;vga_palf_1001:
8218                                <1>      ; cmp     al, 1
8219                                <1>      ; je      short set_overscan_border_color
8220                                <1>      vga_palf_1003:
8221 0000367D 3C03      <1>      cmp     al, 3
8222 0000367F 74B6      <1>      je      short toggle_intensity
8223                                <1>      vga_palf_1007:
8224 00003681 3C07      <1>      cmp     al, 7
8225 00003683 747F      <1>      je      short get_single_palette_reg
8226 00003685 72D5      <1>      jb      short vga_palf_unknown
8227                                <1>      vga_palf_1008:
8228 00003687 3C08      <1>      cmp     al, 8
8229 00003689 7477      <1>      je      short read_overscan_border_color
8230                                <1>      vga_palf_1009:

```

```

8231 0000368B 3C09      <1>      cmp     al, 9
8232                   <1>      ;je      short get_all_palette_reg
8233                   <1>      ; 07/08/2022
8234 0000368D 7505      <1>      jne     short vga_palf_1010
8235 0000368F E966010000 <1>      jmp     get_all_palette_reg
8236                   <1> vga_palf_1010:
8237 00003694 3C10      <1>      cmp     al, 10h
8238                   <1>      ;je      short set_single_dac_reg
8239                   <1>      ; 07/08/2022
8240 00003696 7707      <1>      ja      short vga_palf_1012
8241 00003698 72C2      <1>      jnb     short vga_palf_unknown
8242 0000369A E908010000 <1>      jmp     set_single_dac_reg
8243                   <1> vga_palf_1012:
8244 0000369F 3C12      <1>      cmp     al, 12h
8245                   <1>      ;je      short set_all_dac_reg
8246                   <1>      ; 07/08/2022
8247 000036A1 7707      <1>      ja      short vga_palf_1013
8248 000036A3 72B7      <1>      jnb     short vga_palf_unknown
8249 000036A5 E916010000 <1>      jmp     set_all_dac_reg
8250                   <1> vga_palf_1013:
8251 000036AA 3C13      <1>      cmp     al, 13h
8252                   <1>      ;je      short select_video_dac_color_page
8253                   <1>      ; 07/08/2022
8254 000036AC 7505      <1>      jne     short vga_palf_1015
8255 000036AE E992010000 <1>      jmp     select_video_dac_color_page
8256                   <1> vga_palf_1015:
8257 000036B3 3C15      <1>      cmp     al, 15h
8258                   <1>      ;je      short read_single_dac_reg
8259                   <1>      ; 07/08/2022
8260 000036B5 7707      <1>      ja      short vga_palf_1017
8261 000036B7 72A3      <1>      jnb     short vga_palf_unknown
8262 000036B9 E98D000000 <1>      jmp     read_single_dac_reg
8263                   <1> vga_palf_1017:
8264 000036BE 3C17      <1>      cmp     al, 17h
8265                   <1>      ;je      short read_all_dac_reg
8266                   <1>      ; 07/08/2022
8267 000036C0 7707      <1>      ja      short vga_palf_1018
8268 000036C2 7298      <1>      jnb     short vga_palf_unknown
8269 000036C4 E9A0000000 <1>      jmp     read_all_dac_reg
8270                   <1> vga_palf_1018:
8271 000036C9 3C18      <1>      cmp     al, 18h
8272 000036CB 7464      <1>      je      short set_pel_mask
8273                   <1> vga_palf_1019:
8274 000036CD 3C19      <1>      cmp     al, 19h
8275 000036CF 746C      <1>      je      short read_pel_mask
8276                   <1> vga_palf_101A:
8277 000036D1 3C1A      <1>      cmp     al, 1Ah
8278                   <1>      ;je      short read_video_dac_state
8279                   <1>      ; 07/08/2022
8280 000036D3 758E      <1>      jne     short vga_palf_101B
8281 000036D5 E9AD010000 <1>      jmp     read_video_dac_state
8282                   <1>
8283                   <1> ; 07/08/2022
8284                   <1> set_overscan_border_color:
8285                   <1> ; 10/08/2016
8286                   <1> ; Set Overscan/Border Color Register
8287                   <1> ; BH = 6-bit RGB color to display
8288                   <1> ; for that attribute
8289                   <1>
8290 000036DA B311      <1>      mov     bl, 11h
8291                   <1> ; 07/08/2022
8292                   <1> ;jmp     short set_single_palette_reg
8293                   <1>
8294                   <1> set_single_palette_reg:
8295                   <1> ; 03/08/2022 (TRDOS 386 v2.0.5)
8296                   <1> ; 12/04/2021 (TRDOS 386 v2.0.3, 32 bit push/pop)
8297                   <1> ; 10/08/2016
8298                   <1> ; Set One Palette Register
8299                   <1> ; BL = register number to set
8300                   <1> ; (a 4-bit attribute nibble: 00h-0Fh)
8301                   <1> ; BH = 6-bit RGB color to display
8302                   <1> ; for that attribute
8303                   <1>
8304 000036DC 80FB14      <1>      cmp     bl, 14h
8305                   <1> ;ja      short no_actl_reg1
8306                   <1> ;ja      VIDEO_RETURN
8307                   <1> ; 03/08/2022
8308 000036DF 7605      <1>      jna     short sspr_1
8309 000036E1 E992E4FFFF <1>      jmp     VIDEO_RETURN
8310                   <1> sspr_1:
8311                   <1> ;push    ax
8312                   <1> ;push    dx
8313                   <1> ; 12/04/2021
8314 000036E6 50          <1>      push    eax
8315 000036E7 52          <1>      push    edx
8316 000036E8 66BADA03 <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
8317 000036EC EC          <1>      in      al, dx
8318                   <1> ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
8319                   <1> ; 03/08/2022
8320 000036ED B2C0      <1>      mov     dl, 0C0h
8321 000036EF 88D8      <1>      mov     al, bl
8322 000036F1 EE          <1>      out     dx, al
8323 000036F2 88F8      <1>      mov     al, bh
8324 000036F4 EE          <1>      out     dx, al
8325 000036F5 B020      <1>      mov     al, 20h
8326 000036F7 EE          <1>      out     dx, al
8327                   <1> ; ifdef VBOX
8328                   <1> ;mov     dx, 3DAh ; VGAREG_ACTL_RESET
8329                   <1> ; 03/08/2022
8330 000036F8 B2DA      <1>      mov     dl, 0DAh
8331 000036FA EC          <1>      in      al, dx
8332                   <1> ; endif ; VBOX
8333                   <1> ;pop     dx
8334                   <1> ;pop     ax
8335                   <1> ; 12/04/2021
8336 000036FB 5A          <1>      pop     edx
8337 000036FC 58          <1>      pop     eax
8338                   <1> ;no_actl_reg1:
8339 000036FD E976E4FFFF <1>      jmp     VIDEO_RETURN
8340                   <1>
8341                   <1> ; 07/08/2022
8342                   <1> read_overscan_border_color:
8343                   <1> ; 10/08/2016
8344                   <1> ; Read Overscan Register
8345                   <1> ; OUTPUT:
8346                   <1> ; BH = current rgbRGB value
8347                   <1> ; of the overscan/border register
8348                   <1>
8349 00003702 B311      <1>      mov     bl, 11h
8350                   <1> ; 07/08/2022
8351                   <1> ;jmp     short get_single_palette_reg
8352                   <1>
8353                   <1> get_single_palette_reg:
8354                   <1> ; 03/08/2022

```

```

8355 <1> ; 10/08/2016
8356 <1> ; Read One Palette Register
8357 <1> ; INPUT:
8358 <1> ; BL = Palette register to read (00h-0Fh)
8359 <1> ; OUTPUT:
8360 <1> ; BH = Current rgbRGB value of specified register
8361 <1> ; for that attribute
8362 <1>
8363 00003704 80FB14 <1> cmp bl, 14h
8364 <1> ;ja short no_actl_reg2
8365 <1> ;ja VIDEO_RETURN
8366 <1> ; 03/08/2022
8367 00003707 7605 <1> jna short gspr_1
8368 00003709 E96AE4FFFF <1> jmp VIDEO_RETURN
8369 <1> gspr_1:
8370 0000370E 66BADA03 <1> mov dx, 3DAh ; VGAREG_ACTL_RESET
8371 <1> ; 03/08/2022
8372 00003712 B2DA <1> mov dl, 0DAh
8373 00003714 EC <1> in al, dx
8374 <1> ;mov dx, 3C0h ; VGAREG_ACTL_ADDRESS
8375 <1> ; 03/08/2022
8376 00003715 B2C0 <1> mov dl, 0C0h
8377 00003717 88D8 <1> mov al, bl
8378 00003719 EE <1> out dx, al
8379 <1> ;mov dx, 3C1h ; VGAREG_ACTL_READ_DATA
8380 <1> ; 03/08/2022
8381 0000371A FEC2 <1> inc dl ; dx = 3C1h
8382 0000371C EC <1> in al, dx
8383 0000371D 8844240D <1> mov [esp+13], al ; bh
8384 <1> ;mov dx, 3DAh ; VGAREG_ACTL_RESET
8385 <1> ; 03/08/2022
8386 00003721 B2DA <1> mov dl, 0DAh
8387 00003723 EC <1> in al, dx
8388 <1> ;mov dx, 3C0h ; VGAREG_ACTL_ADDRESS
8389 <1> ; 03/08/2022
8390 00003724 B2C0 <1> mov dl, 0C0h
8391 00003726 B020 <1> mov al, 20h
8392 00003728 EE <1> out dx, al
8393 <1> ; ifdef VBOX
8394 <1> ;mov dx, 3DAh ; VGAREG_ACTL_RESET
8395 <1> ; 03/08/2022
8396 00003729 B2DA <1> mov dl, 0DAh
8397 0000372B EC <1> in al, dx
8398 <1> ; endif ; VBOX
8399 0000372C E947E4FFFF <1> jmp VIDEO_RETURN
8400 <1>
8401 <1> set_pel_mask:
8402 <1> ; 10/08/2016
8403 <1> ; BL = mask value
8404 00003731 66BAC603 <1> mov dx, 3C6h ; VGAREG_PEL_MASK
8405 00003735 88D8 <1> mov al, bl
8406 00003737 EE <1> out dx, al
8407 00003738 E93BE4FFFF <1> jmp VIDEO_RETURN
8408 <1>
8409 <1> read_pel_mask:
8410 <1> ; 10/08/2016
8411 <1> ; Output: BL = mask value
8412 0000373D 66BAC603 <1> mov dx, 3C6h ; VGAREG_PEL_MASK
8413 00003741 EC <1> in al, dx
8414 00003742 8844240C <1> mov [esp+12], al ; bl
8415 00003746 E92DE4FFFF <1> jmp VIDEO_RETURN
8416 <1>
8417 <1> read_single_dac_reg:
8418 <1> ; 02/08/2022
8419 <1> ; 10/08/2016
8420 <1> ; Read One DAC Color Register
8421 <1> ; INPUT:
8422 <1> ; BX = color register to read (0-255)
8423 <1> ; OUTPUT:
8424 <1> ; CH = green value (00h-3Fh)
8425 <1> ; CL = blue value (00h-3Fh)
8426 <1> ; DH = red value (00h-3Fh)
8427 <1>
8428 0000374B 66BAC703 <1> mov dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
8429 0000374F 88D8 <1> mov al, bl
8430 00003751 EE <1> out dx, al
8431 <1> ;mov dx, 3C9h ; VGAREG_DAC_DATA
8432 <1> ; 02/08/2022
8433 00003752 B2C9 <1> mov dl, 0C9h
8434 00003754 EC <1> in al, dx
8435 00003755 88442415 <1> mov [esp+21], al ; dh
8436 00003759 EC <1> in al, dx
8437 0000375A 88C5 <1> mov ch, al
8438 0000375C EC <1> in al, dx
8439 0000375D 88C1 <1> mov cl, al
8440 0000375F 66894C2410 <1> mov [esp+16], cx ; cx
8441 00003764 E90FE4FFFF <1> jmp VIDEO_RETURN
8442 <1>
8443 <1> read_all_dac_reg:
8444 <1> ; 02/08/2022
8445 <1> ; 12/08/2016
8446 <1> ; 11/08/2016
8447 <1> ; 10/08/2016
8448 <1> ; Read a Block of DAC Color Registers
8449 <1> ; BX = first DAC register to read (0-00FFh)
8450 <1> ; ECX = number of registers to read (0-00FFh)
8451 <1> ; EDX = addr of a buffer to hold R,G,B values
8452 <1> ; (CX*3 bytes long)
8453 <1>
8454 00003769 89D7 <1> mov edi, edx ; user buffer
8455 0000376B 89CA <1> mov edx, ecx
8456 <1> ;shl dx, 1 ; *2
8457 <1> ; 02/08/2022
8458 0000376D D1E2 <1> shl edx, 1
8459 0000376F 01CA <1> add edx, ecx ; edx = 3*ecx
8460 00003771 89E5 <1> mov ebp, esp
8461 00003773 89EE <1> mov esi, ebp
8462 00003775 29D6 <1> sub esi, edx
8463 00003777 6683E6FC <1> and si, 0FFFCh ; (dword alignment)
8464 0000377B 89F4 <1> mov esp, esi
8465 0000377D 52 <1> push edx ; 3*ecx
8466 0000377E 66BAC703 <1> mov dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
8467 00003782 88D8 <1> mov al, bl
8468 00003784 EE <1> out dx, al
8469 00003785 66BAC903 <1> mov dx, 3C9h ; VGAREG_DAC_DATA
8470 00003789 89F3 <1> mov ebx, esi
8471 <1> read_dac_loop:
8472 0000378B EC <1> in al, dx
8473 0000378C 8803 <1> mov [ebx], al
8474 0000378E 43 <1> inc ebx
8475 0000378F EC <1> in al, dx
8476 00003790 8803 <1> mov [ebx], al
8477 00003792 43 <1> inc ebx
8478 00003793 EC <1> in al, dx

```

```

8479 00003794 8803      <1>      mov     [ebx], al
8480 00003796 43      <1>      inc     ebx
8481      <1>      ;dec     cx
8482      <1>      ; 02/08/2022
8483 00003797 49      <1>      dec     ecx
8484 00003798 75F1     <1>      jnz     short read_dac_loop
8485 0000379A 59      <1>      pop     ecx ; 3*ecx
8486      <1>      ; ECX = transfer (byte) count
8487      <1>      ; ESI = source address in system space
8488      <1>      ; EDI = user's buffer address
8489 0000379B E86BD30000 <1>      call    transfer_to_user_buffer
8490 000037A0 89EC     <1>      mov     esp, ebp
8491 000037A2 E9D1E3FFFF <1>      jmp     VIDEO_RETURN
8492      <1>
8493      <1> set_single_dac_reg:
8494      <1>      ; 03/08/2022 (TRDOS 386 v2.0.5)
8495      <1>      ; 12/04/2021 (TRDOS 386 v2.0.3, 32 bit push/pop)
8496      <1>      ; 10/08/2016
8497      <1>      ; Set One DAC Color Register
8498      <1>      ; BX = color register to set (0-255)
8499      <1>      ; CH = green value (00h-3Fh)
8500      <1>      ; CL = blue value (00h-3Fh)
8501      <1>      ; DH = red value (00h-3Fh)
8502      <1>
8503      <1>      ;push    dx
8504      <1>      ; 12/04/2021
8505 000037A7 52      <1>      push   edx
8506 000037A8 66BAC803 <1>      mov     dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
8507 000037AC 88D8     <1>      mov     al, bl
8508 000037AE EE      <1>      out     dx, al
8509      <1>      ;;mov    dx, 3C9h ; VGAREG_DAC_DATA
8510      <1>      ;inc     dx
8511      <1>      ; 03/08/2022
8512 000037AF FEC2     <1>      inc     dl
8513      <1>      ;pop     ax
8514      <1>      ; 12/04/2021
8515 000037B1 58      <1>      pop     eax
8516 000037B2 88E0     <1>      mov     al, ah
8517 000037B4 EE      <1>      out     dx, al
8518 000037B5 88E8     <1>      mov     al, ch
8519 000037B7 EE      <1>      out     dx, al
8520 000037B8 88C8     <1>      mov     al, cl
8521 000037BA EE      <1>      out     dx, al
8522 000037BB E9B8E3FFFF <1>      jmp     VIDEO_RETURN
8523      <1>
8524      <1> set_all_dac_reg:
8525      <1>      ; 02/08/2022
8526      <1>      ; 12/08/2016
8527      <1>      ; 11/08/2016
8528      <1>      ; 10/08/2016
8529      <1>      ; Set a Block of DAC Color Register
8530      <1>      ; BX = first DAC register to set (0-00FFh)
8531      <1>      ; ECX = number of registers to set (0-00FFh)
8532      <1>      ; EDX = addr of a table of R,G,B values
8533      <1>      ; (it will be CX*3 bytes long)
8534      <1>
8535 000037C0 89D6     <1>      mov     esi, edx ; user buffer
8536 000037C2 89CA     <1>      mov     edx, ecx
8537      <1>      ;shl     cx, 1 ; *2
8538      <1>      ; 02/08/2022
8539 000037C4 D1E1     <1>      shl     ecx, 1
8540 000037C6 01D1     <1>      add     ecx, edx ; ecx = 3*ecx
8541 000037C8 89E5     <1>      mov     ebp, esp
8542 000037CA 89EF     <1>      mov     edi, ebp
8543 000037CC 29CF     <1>      sub     edi, ecx
8544 000037CE 6683E7FC <1>      and     di, 0FFFCh ; (dword alignment)
8545 000037D2 89FC     <1>      mov     esp, edi
8546 000037D4 E87CD30000 <1>      call    transfer_from_user_buffer
8547      <1>      ;jc     VIDEO_RETURN
8548      <1>
8549 000037D9 89D1     <1>      mov     ecx, edx
8550 000037DB 66BAC803 <1>      mov     dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
8551 000037DF 88D8     <1>      mov     al, bl
8552 000037E1 EE      <1>      out     dx, al
8553      <1>      ;mov    dx, 3C9h ; VGAREG_DAC_DATA
8554      <1>      ; 02/08/2022
8555 000037E2 FEC2     <1>      inc     dl
8556      <1> set_dac_loop:
8557 000037E4 8A07     <1>      mov     al, [edi]
8558 000037E6 EE      <1>      out     dx, al
8559 000037E7 47      <1>      inc     edi
8560 000037E8 8A07     <1>      mov     al, [edi]
8561 000037EA EE      <1>      out     dx, al
8562 000037EB 47      <1>      inc     edi
8563 000037EC 8A07     <1>      mov     al, [edi]
8564 000037EE EE      <1>      out     dx, al
8565 000037EF 47      <1>      inc     edi
8566      <1>      ;dec     cx
8567      <1>      ; 02/08/2022
8568 000037F0 49      <1>      dec     ecx
8569 000037F1 75F1     <1>      jnz     short set_dac_loop
8570 000037F3 89EC     <1>      mov     esp, ebp
8571 000037F5 E97EE3FFFF <1>      jmp     VIDEO_RETURN
8572      <1>
8573      <1> get_all_palette_reg:
8574      <1>      ; 03/08/2022
8575      <1>      ; 10/08/2016
8576      <1>      ; Read All Palette Registers
8577      <1>      ; EDX = Address of 17-byte buffer
8578      <1>      ; to receive data
8579      <1>
8580 000037FA 89D7     <1>      mov     edi, edx
8581 000037FC 89E3     <1>      mov     ebx, esp
8582 000037FE 89DE     <1>      mov     esi, ebx
8583 00003800 83EC14   <1>      sub     esp, 20
8584      <1>
8585 00003803 B100     <1>      mov     cl, 0
8586      <1> get_palette_loop:
8587 00003805 66BADA03 <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
8588 00003809 EC      <1>      in      al, dx
8589      <1>      ;mov    dx, 3C0h ; VGAREG_ACTL_ADDRESS
8590      <1>      ; 03/08/2022
8591 0000380A B2C0     <1>      mov     dl, 0C0h
8592 0000380C 88C8     <1>      mov     al, cl
8593 0000380E EE      <1>      out     dx, al
8594      <1>      ;mov    dx, 3C1h ; VGAREG_ACTL_READ_DATA
8595      <1>      ; 03/08/2022
8596      <1>      ;mov    dl, 0C1h
8597 0000380F FEC2     <1>      inc     dl
8598 00003811 EC      <1>      in      al, dx
8599 00003812 8803     <1>      mov     [ebx], al
8600 00003814 43      <1>      inc     ebx
8601 00003815 FEC1     <1>      inc     cl
8602 00003817 80F910     <1>      cmp     cl, 10h

```

```

8603 0000381A 75E9      <1>     jne     short get_palette_loop
8604                      <1>     ;mov     dx, 3DAh ; VGAREG_ACTL_RESET
8605                      <1>     ; 03/08/2022
8606 0000381C B2DA      <1>     mov     dl, 0DAh
8607 0000381E EC         <1>     in      al, dx
8608                      <1>     ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
8609                      <1>     ; 03/08/2022
8610 0000381F B2C0      <1>     mov     dl, 0C0h
8611 00003821 B011      <1>     mov     al, 11h
8612 00003823 EE         <1>     out     dx, al
8613                      <1>     ;mov     dx, 3C1h ; VGAREG_ACTL_READ_DATA
8614                      <1>     ; 03/08/2022
8615 00003824 FEC2      <1>     inc     dl ; dx = 3C1h
8616 00003826 EC         <1>     in      al, dx
8617 00003827 8803      <1>     mov     [ebx], al
8618                      <1>     ;mov     dx, 3DAh ; VGAREG_ACTL_RESET
8619                      <1>     ; 03/08/2022
8620 00003829 B2DA      <1>     mov     dl, 0DAh
8621 0000382B EC         <1>     in      al, dx
8622                      <1>     ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
8623                      <1>     ; 03/08/2022
8624 0000382C B2C0      <1>     mov     dl, 0C0h
8625 0000382E B020      <1>     mov     al, 20h
8626 00003830 EE         <1>     out     dx, al
8627                      <1>     ; ifdef VBOX
8628                      <1>     ;mov     dx, 3DAh ; VGAREG_ACTL_RESET
8629                      <1>     ; 03/08/2022
8630 00003831 B2DA      <1>     mov     dl, 0DAh
8631 00003833 EC         <1>     in      al, dx
8632                      <1>     ; endif ; VBOX
8633                      <1>
8634                      <1>     ;mov     ecx, 17 ; transfer (byte) count
8635                      <1>     ; 03/08/2022
8636 00003834 29C9      <1>     sub     ecx, ecx
8637 00003836 B111      <1>     mov     cl, 17
8638                      <1>
8639                      <1>     ; ESI = source address in system space
8640                      <1>     ; EDI = user's buffer address
8641 00003838 E8CED20000 <1>     call    transfer_to_user_buffer
8642                      <1>
8643 0000383D 83C414      <1>     add     esp, 20
8644 00003840 E933E3FFFF <1>     jmp     VIDEO_RETURN
8645                      <1>
8646                      <1> select_video_dac_color_page:
8647                      <1>     ; 02/08/2022 (TRDOS 386 v2.0.5, code optimization)
8648                      <1>     ; 12/04/2021 (TRDOS 386 v2.0.3, 32 bit push/pop)
8649                      <1>     ; 10/08/2016
8650                      <1>     ; DAC Color Paging Functions
8651                      <1>     ; BL = 00H = select color paging mode
8652                      <1>     ;         ; BH = paging mode
8653                      <1>     ;         ; 00h = 4 blocks of 64 registers
8654                      <1>     ;         ; 01h = 16 blocks of 16 registers
8655                      <1>     ; BL = 01H = activate color page
8656                      <1>     ;         ; BH = DAC color page number
8657                      <1>     ;         ; 00h-03h (4-page/64-reg mode)
8658                      <1>     ;         ; 00h-0Fh (16-page/16-reg mode)
8659                      <1>
8660 00003845 66BADA03    <1>     mov     dx, 3DAh ; VGAREG_ACTL_RESET
8661 00003849 EC         <1>     in      al, dx
8662                      <1>     ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
8663                      <1>     ; 02/08/2022
8664 0000384A B240      <1>     mov     dl, 40h
8665 0000384C B010      <1>     mov     al, 10h
8666 0000384E EE         <1>     out     dx, al
8667                      <1>     ;mov     dx, 3C1h ; VGAREG_ACTL_READ_DATA
8668                      <1>     ; 02/08/2022
8669 0000384F FEC2      <1>     inc     dl ; mov dl, 0C1h
8670 00003851 EC         <1>     in      al, dx
8671 00003852 80E301      <1>     and     bl, 01h
8672 00003855 750C      <1>     jnz     short set_dac_page
8673 00003857 247F      <1>     and     al, 07Fh
8674 00003859 C0E707      <1>     shl     bh, 7
8675 0000385C 08F8      <1>     or      al, bh
8676                      <1>     ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
8677                      <1>     ; 02/08/2022
8678 0000385E FECA      <1>     dec     dl ; mov dl, 0C0h
8679 00003860 EE         <1>     out     dx, al
8680 00003861 EB19      <1>     jmp     short set_actl_normal
8681                      <1> set_dac_page:
8682                      <1>     ;push     ax
8683                      <1>     ; 12/04/2021
8684 00003863 50         <1>     push    eax
8685 00003864 66BADA03    <1>     mov     dx, 3DAh ; VGAREG_ACTL_RESET
8686 00003868 EC         <1>     in      al, dx
8687                      <1>     ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
8688                      <1>     ; 02/08/2022
8689 00003869 B2C0      <1>     mov     dl, 0C0h
8690 0000386B B014      <1>     mov     al, 14h
8691 0000386D EE         <1>     out     dx, al
8692                      <1>     ;pop      ax
8693                      <1>     ; 12/04/2021
8694 0000386E 58         <1>     pop     eax
8695 0000386F 2480      <1>     and     al, 80h
8696 00003871 7503      <1>     jnz     short set_dac_16_page
8697 00003873 C0E702      <1>     shl     bh, 2
8698                      <1> set_dac_16_page:
8699                      <1>     and     bh, 0Fh
8700 00003879 88F8      <1>     mov     al, bh
8701 0000387B EE         <1>     out     dx, al
8702                      <1> set_actl_normal:
8703 0000387C B020      <1>     mov     al, 20h
8704 0000387E EE         <1>     out     dx, al
8705                      <1>     ; ifdef VBOX
8706                      <1>     ;mov     dx, 3DAh ; VGAREG_ACTL_RESET
8707                      <1>     ; 02/08/2022
8708 0000387F B2DA      <1>     mov     dl, 0DAh
8709 00003881 EC         <1>     in      al, dx
8710                      <1>     ; endif ; VBOX
8711 00003882 E9F1E2FFFF <1>     jmp     VIDEO_RETURN
8712                      <1>
8713                      <1> read_video_dac_state:
8714                      <1>     ; 10/08/2016
8715                      <1>     ; Query DAC Color Paging State
8716                      <1>     ; Output:
8717                      <1>     ; BH = current active DAC color page
8718                      <1>     ; BL = current active DAC paging mode
8719                      <1>
8720 00003887 66BADA03    <1>     mov     dx, 3DAh ; VGAREG_ACTL_RESET
8721 0000388B EC         <1>     in      al, dx
8722 0000388C 66BAC003    <1>     mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
8723 00003890 B010      <1>     mov     al, 10h
8724 00003892 EE         <1>     out     dx, al
8725 00003893 66BAC103    <1>     mov     dx, 3C1h ; VGAREG_ACTL_READ_DATA
8726 00003897 EC         <1>     in      al, dx

```



```

8727 00003898 88C3      <1>      mov     bl, al
8728 0000389A C0EB07      <1>      shr     bl, 7
8729 0000389D 66BADA03    <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
8730 000038A1 EC          <1>      in      al, dx
8731 000038A2 66BAC003    <1>      mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
8732 000038A6 B014      <1>      mov     al, 14h
8733 000038A8 EE          <1>      out     dx, al
8734 000038A9 66BAC103    <1>      mov     dx, 3C1h ; VGAREG_ACTL_READ_DATA
8735 000038AD EC          <1>      in      al, dx
8736 000038AE 88C7      <1>      mov     bh, al
8737 000038B0 80E70F      <1>      and     bh, 0Fh
8738 000038B3 F6C301      <1>      test    bl, 01
8739 000038B6 7503      <1>      jnz     short get_dac_16_page
8740 000038B8 C0EF02      <1>      shr     bh, 2
8741          <1> get_dac_16_page:
8742 000038BB 66BADA03    <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
8743 000038BF EC          <1>      in      al, dx
8744 000038C0 66BAC003    <1>      mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
8745 000038C4 B020      <1>      mov     al, 20h
8746 000038C6 EE          <1>      out     dx, al
8747          <1> ; ifdef VBOX
8748 000038C7 66BADA03    <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
8749 000038CB EC          <1>      in      al, dx
8750          <1> ; endif ; VBOX
8751 000038CC 66895C240C <1>      mov     [esp+12], bx ; bx
8752 000038D1 E9A2E2FFFF <1>      jmp     VIDEO_RETURN
8753          <1>
8754          <1> ; 23/11/2020 - TRDOS 386 v2.0.3
8755          <1> ; VBE 2 BOCHS/QEMU emulator extensions
8756          <1> ; for TRDOS 386 v2 kernel (video bios)
8757          <1>
8758          <1> ; BOCH/QEMU VBE2 VGA BIOS code
8759          <1> ; by Jeroen Janssen (2002)
8760          <1> ; by Volker Rupper (2003-2020)
8761          <1> ; vbe.c (02/01/2020)
8762          <1>
8763          <1> ; vbe.h (02/01/2020)
8764          <1>
8765          <1> VBE_DISPI_BANK_ADDRESS      equ     0A0000h
8766          <1> VBE_DISPI_BANK_SIZE_KB      equ     64
8767          <1>
8768          <1> VBE_DISPI_MAX_XRES equ     2560
8769          <1> VBE_DISPI_MAX_YRES equ     1600
8770          <1>
8771          <1> VBE_DISPI_IOPORT_INDEX      equ     01CEh
8772          <1> VBE_DISPI_IOPORT_DATA      equ     01CFh
8773          <1>
8774          <1> VBE_DISPI_INDEX_ID equ     00h
8775          <1> VBE_DISPI_INDEX_XRES      equ     01h
8776          <1> VBE_DISPI_INDEX_YRES      equ     02h
8777          <1> VBE_DISPI_INDEX_BPP       equ     03h
8778          <1> VBE_DISPI_INDEX_ENABLE    equ     04h
8779          <1> VBE_DISPI_INDEX_BANK      equ     05h
8780          <1> VBE_DISPI_INDEX_VIRT_WIDTH equ    06h
8781          <1> VBE_DISPI_INDEX_VIRT_HEIGHT equ   07h
8782          <1> VBE_DISPI_INDEX_X_OFFSET   equ    08h
8783          <1> VBE_DISPI_INDEX_Y_OFFSET   equ    09h
8784          <1> VBE_DISPI_INDEX_VIDEO_MEMORY_64K equ 0Ah
8785          <1> VBE_DISPI_INDEX_DDC      equ     0Bh
8786          <1>
8787          <1> VBE_DISPI_ID0              equ     0B0C0h
8788          <1> VBE_DISPI_ID1              equ     0B0C1h
8789          <1> VBE_DISPI_ID2              equ     0B0C2h
8790          <1> VBE_DISPI_ID3              equ     0B0C3h
8791          <1> VBE_DISPI_ID4              equ     0B0C4h
8792          <1> VBE_DISPI_ID5              equ     0B0C5h
8793          <1>
8794          <1> VBE_DISPI_DISABLED equ     00h
8795          <1> VBE_DISPI_ENABLED equ     01h
8796          <1> VBE_DISPI_GETCAPS equ     02h
8797          <1> VBE_DISPI_8BIT_DAC equ     20h
8798          <1> VBE_DISPI_LFB_ENABLED      equ     40h
8799          <1> VBE_DISPI_NOCLEARMEM      equ     80h
8800          <1>
8801          <1> VBE_DISPI_LFB_PHYSICAL_ADDRESS equ 0E0000000h
8802          <1>
8803          <1> ; ***
8804          <1>
8805          <1> ;// VBE Return Status Info
8806          <1> ;// AL
8807          <1> VBE_RETURN_STATUS_SUPPORTED      equ     4Fh
8808          <1> VBE_RETURN_STATUS_UNSUPPORTED    equ     00h
8809          <1> ;// AH
8810          <1> VBE_RETURN_STATUS_SUCCESSFULL    equ     00h
8811          <1> VBE_RETURN_STATUS_FAILED          equ     01h
8812          <1> VBE_RETURN_STATUS_NOT_SUPPORTED    equ     02h
8813          <1> VBE_RETURN_STATUS_INVALID          equ     03h
8814          <1>
8815          <1> ;// VBE Mode Numbers
8816          <1>
8817          <1> VBE_MODE_VESA_DEFINED      equ     0100h
8818          <1> VBE_MODE_REFRESH_RATE_USE_CRTC equ     0800h
8819          <1> VBE_MODE_LINEAR_FRAME_BUFFER      equ     4000h
8820          <1> VBE_MODE_PRESERVE_DISPLAY_MEMORY equ     8000h
8821          <1>
8822          <1> ;// Mode Attributes
8823          <1>
8824          <1> VBE_MODE_ATTRIBUTE_SUPPORTED      equ     0001h
8825          <1> VBE_MODE_ATTRIBUTE_EXTENDED_INFO_AVAILABLE equ     0002h
8826          <1> VBE_MODE_ATTRIBUTE_COLOR_MODE    equ     0008h
8827          <1> VBE_MODE_ATTRIBUTE_GRAPHICS_MODE      equ     0010h
8828          <1> VBE_MODE_ATTRIBUTE_LINEAR_FRAME_BUFFER_MODE equ     0080h
8829          <1> VBE_MODE_ATTRIBUTE_DOUBLE_SCAN_MODE      equ     0100h
8830          <1> VBE_MODE_ATTRIBUTE_INTERLACE_MODE      equ     0200h
8831          <1>
8832          <1> ;// window attributes
8833          <1>
8834          <1> VBE_WINDOW_ATTRIBUTE_RELOCATABLE      equ     01h
8835          <1> VBE_WINDOW_ATTRIBUTE_READABLE      equ     02h
8836          <1> VBE_WINDOW_ATTRIBUTE_WRITEABLE      equ     04h
8837          <1>
8838          <1> ;/* video memory */
8839          <1> VGAMEM_GRAPH equ 0A000h
8840          <1> VGAMEM_CTEXT equ 0B800h
8841          <1> ;VGAMEM_MTEXT equ 0B000h
8842          <1>
8843          <1> ;// Memory model
8844          <1>
8845          <1> ;VBE_MEMORYMODEL_TEXT_MODE equ     00h
8846          <1> ;VBE_MEMORYMODEL_CGA_GRAPHICS      equ     01h
8847          <1> ;VBE_MEMORYMODEL_PLANAR            equ     03h
8848          <1> VBE_MEMORYMODEL_PACKED_PIXEL      equ     04h
8849          <1> ;VBE_MEMORYMODEL_NON_CHAIN_4_256    equ     05h
8850          <1> VBE_MEMORYMODEL_DIRECT_COLOR      equ     06h

```

```

8851 <1> ;VBE_MEMORYMODEL_YUV equ 07h
8852 <1>
8853 <1> ;// DirectColorModeInfo
8854 <1>
8855 <1> ;VBE_DIRECTCOLOR_COLOR_RAMP_PROGRAMMABLE equ 01h
8856 <1> VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE equ 02h
8857 <1>
8858 <1> VBE_DISPI_TOTAL_VIDEO_MEMORY_MB equ 16
8859 <1>
8860 <1> ; 24/11/2020
8861 <1> ; vbe.c
8862 <1>
8863 <1> %if 1
8864 <1>
8865 <1> _vbe_biosfn_return_mode_info:
8866 <1> ; 15/12/2020
8867 <1> ; 12/12/2020
8868 <1> ; Return VBE Mode Information
8869 <1> ; (call from 'sysvideo')
8870 <1> ;
8871 <1> ; Input:
8872 <1> ; cx = video (bios) mode
8873 <1> ; Output:
8874 <1> ; cf = 0 -> (successful)
8875 <1> ; MODE_INFO_LIST addr contains MODEINFO
8876 <1> ; cf = 1 -> error
8877 <1> ;
8878 <1> ; Modified registers: eax, edx, edi
8879 <1> ;
8880 <1>
8881 <1> ; pushes for subroutine stack pops compatibility
8882 <1>
8883 <1> ;push ds ; *
8884 <1> ;push es ; **
8885 <1>
8886 000038D6 55 <1> push ebp ; ***
8887 000038D7 56 <1> push esi ; ****
8888 <1>
8889 000038D8 31FF <1> xor edi, edi ; mov edi, 0
8890 <1>
8891 000038DA 803D[86090000]03 <1> cmp byte [vbe3], 3
8892 000038E1 7221 <1> jb short _vbe_rmi_1
8893 <1>
8894 <1> ;sub edi, edi ; 0 = kernel call (sign)
8895 <1> ; ; no transfer to user's buffer
8896 <1>
8897 <1> ; cx = Video mode (for 4F01h, with LFB flag)
8898 <1>
8899 000038E3 66B8014F <1> mov ax, 4F01h
8900 <1>
8901 000038E7 E826E0FFFF <1> call _vbe3_pmf_n_return_mode_info
8902 <1>
8903 000038EC 6683F84F <1> cmp ax, 004Fh
8904 000038F0 7533 <1> jne short _vbe_rmi_2 ; fail
8905 <1>
8906 <1> ; 15/12/2020
8907 <1> ; cx = vbe video mode
8908 000038F2 80E501 <1> and ch, 01h ; clear LFB flag
8909 000038F5 BEFE7B0900 <1> mov esi, VBE3MODEINFOBLOCK - 2
8910 000038FA 66890E <1> mov [esi], cx ; MODEINFO.mode
8911 000038FD E8A5000000 <1> call set_lfbinfo_table
8912 00003902 EB22 <1> jmp short _vbe_rmi_3 ; cf = 0
8913 <1> _vbe_rmi_1:
8914 00003904 803D[86090000]02 <1> cmp byte [vbe3], 2
8915 0000390B 7219 <1> jb short _vbe_rmi_3 ; cf = 1
8916 0000390D A0[87090000] <1> mov al, [vbe2bios] ; 0C0h-0C5h for emu (*)
8917 00003912 3CC0 <1> cmp al, 0C0h ; BOCHS/QEMU/VIRTUALBOX (*) ?
8918 00003914 7210 <1> jb short _vbe_rmi_3 ; cf = 1
8919 00003916 3CC5 <1> cmp al, 0C5h ; (*)
8920 00003918 770B <1> ja short _vbe_rmi_2 ; unknown vbios !?
8921 <1>
8922 <1> ;xor edi, edi ; 0 = kernel call (sign)
8923 <1> ; ; no transfer to user's buffer
8924 <1>
8925 <1> ;mov ax, 4F01h
8926 <1>
8927 <1> ; cx = Video mode (for 4F01h, with LFB flag)
8928 <1>
8929 0000391A E80A000000 <1> call vbe_biosfn_return_mode_info
8930 0000391F 6683F84F <1> cmp ax, 004Fh ; successful ?
8931 00003923 7401 <1> je short _vbe_rmi_3 ; cf = 0
8932 <1> _vbe_rmi_2:
8933 00003925 F9 <1> stc
8934 <1> ; cf = 1
8935 <1> _vbe_rmi_3:
8936 00003926 5E <1> pop esi ; ****
8937 00003927 5D <1> pop ebp ; ***
8938 <1>
8939 <1> ;pop es ; **
8940 <1> ;pop ss ; *
8941 <1>
8942 00003928 C3 <1> retn
8943 <1>
8944 <1>
8945 <1> ; * (TRDOS 386, INT 31h, VESA Video Bios functions)
8946 <1> ; * -----
8947 <1> ; * Function 01h - Return VBE Mode Information
8948 <1> ; * -----
8949 <1> ; * Input:
8950 <1> ; * AX = 4F01h
8951 <1> ; * CX = Mode number
8952 <1> ; * (ES:DI) EDI = Pointer to ModeInfoBlock structure
8953 <1> ; * Output:
8954 <1> ; * AX = VBE Return Status
8955 <1> ; * -----
8956 <1> ; * -----
8957 <1> ; *
8958 <1>
8959 <1> vbe_biosfn_return_mode_info:
8960 <1> ; 03/08/2022 (TRDOS 386 v2.0.5)
8961 <1> ; 15/12/2020
8962 <1> ; 14/12/2020
8963 <1> ; 12/12/2020
8964 <1> ; 11/12/2020 (TRDOS 386 v2.0.3)
8965 <1> ;
8966 <1> ; Input:
8967 <1> ; cx = video (bios) mode
8968 <1> ; edi = ModeInfoBlock buffer address
8969 <1> ; (in user's memory space)
8970 <1> ; (ax = 4F01h)
8971 <1> ; Output:
8972 <1> ; ax = 004Fh (successful)
8973 <1> ; ah > 0 -> error
8974 <1> ;

```

```

8975      <1>      ; Modified registers: esi
8976      <1>
8977      <1>      ;;push ds ; *
8978      <1>      ;;push es ; **
8979      <1>      ;;push ebp ; ***
8980      <1>      ;;push esi ; ****
8981      <1>
8982      <1>      test ch, 1
8983      <1>      jnz short vbe_rmi_1
8984      <1>
8985      <1>      ; mode number < 100h
8986      <1>      ; CGA/VGA mode is not proper this VBE function
8987      <1>
8988      <1>      sub eax, eax
8989      <1>      vbe_rmi_0:
8990      <1>      ;mov ax, 0100h ; Function is not supported
8991      <1>      mov ah, 1
8992      <1>      retn
8993      <1>      vbe_rmi_1:
8994      <1>      push edx ; *****
8995      <1>      push ecx ; *****
8996      <1>      push ebx ; *****
8997      <1>      push edi ; *****
8998      <1>
8999      <1>      ; 14/12/2020
9000      <1>      mov ebx, ecx
9001      <1>
9002      <1>      ;xor eax, eax
9003      <1>      and bh, 0C1h ; use bit 15, 14, 8 only (for bh)
9004      <1>      mov [vbe_mode_x], bh
9005      <1>      ;and bx, 1FFh
9006      <1>      and bh, 1
9007      <1>      mov bh, 1
9008      <1>
9009      <1>      ; Alternative 2 (instead of 'Mode_info_find_mode')
9010      <1>      call set_mode_info_list ; (alternative 2)
9011      <1>
9012      <1>      ; eax = 0
9013      <1>
9014      <1>      ;mov bx, [esi] ; mode
9015      <1>
9016      <1>      ; Alternative 1 (instead of 'set_mode_info_list')
9017      <1>      ;call mode_info_find_mode ; (alternative 1)
9018      <1>
9019      <1>      or esi, esi
9020      <1>      ; 14/12/2020
9021      <1>      jz short vbe_rmi_4 ; VBE mode number is wrong
9022      <1>      ; or it is not supported
9023      <1>
9024      <1>      ; 15/12/2020
9025      <1>      ;mov bx, [esi] ; mode
9026      <1>
9027      <1>      ; 12/12/2020
9028      <1>      ;call set_lfbinfo_table
9029      <1>
9030      <1>      test byte [vbe_mode_x], 40h ; LFB model ?
9031      <1>      jz short vbe_rmi_2
9032      <1>
9033      <1>      mov byte [esi+MODEINFO.NumberOfBanks], 1
9034      <1>      vbe_rmi_2:
9035      <1>      ; (vbe.c, 02/01/2020, vruppert)
9036      <1>      ; 11/12/2020 (Erdogan Tan, video.s)
9037      <1>      ; Bochs Graphics Adapter
9038      <1>      ; vendor_id: 1111h, device id: 1234h
9039      <1>
9040      <1>      call pci_get_lfb_addr
9041      <1>      ;or eax, eax
9042      <1>      jz short vbe_rmi_3
9043      <1>      ; zf = 0, ax > 0 (high word of LFB address)
9044      <1>      ; set/change LFB address in MODEINFO structure
9045      <1>      mov [esi+MODEINFO.PhysBasePtr+2], ax
9046      <1>      ; 12/12/2020
9047      <1>      ;mov [edi+LFBINFO.LFB_addr+2], ax
9048      <1>      vbe_rmi_3:
9049      <1>      ;test byte [esi+MODEINFO.winAAttributes], 1
9050      <1>      ; ; VBE_WINDOW_ATTRIBUTE_RELOCATABLE = 1
9051      <1>      ; ; jz short vbe_rmi_4
9052      <1>      ; ; 11/12/2020
9053      <1>      ; ; In fact, this is far call address in (Bochs/BGA) Video Bios
9054      <1>      ; ; Direct user access to kernel subroutines is not possible
9055      <1>      ; ; in TRDOS 386. Also, TRDOS 386 kernel will support only LFB.
9056      <1>      ; ; Bank select may be a separate sysvideo function in future
9057      <1>      ; ; (if it will be required).
9058      <1>      ;mov dword [esi+MODEINFO.winFuncPtr], dispi_set_bank_farcall
9059      <1>      ;vbe_rmi_4:
9060      <1>      ; 12/12/2020
9061      <1>      call set_lfbinfo_table
9062      <1>
9063      <1>      ; 11/12/2020
9064      <1>      ; copy 68 bytes of MODE_INFO_LIST to user
9065      <1>
9066      <1>      mov edi, [esp] ; user's buffer address
9067      <1>      ; 12/12/2020
9068      <1>      or edi, edi ; 0 = kernel call
9069      <1>      ; ; (call from '_vbe_biosfn_return_mode_info')
9070      <1>      jz short vbe_rmi_6
9071      <1>
9072      <1>      ; 15/12/2020
9073      <1>      ; prepare 256 bytes MODEINFO buffer at VBE3MODEINFOBLOCK
9074      <1>      ; and then, copy buffer content to user's buffer
9075      <1>      push edi
9076      <1>      mov esi, MODE_INFO_LIST + 2 ; MODEINFO.ModeAttributes
9077      <1>      mov edi, VBE3MODEINFOBLOCK
9078      <1>      ;mov ecx, 66/4 ; 66 bytes
9079      <1>      ; 03/08/2022
9080      <1>      sub ecx, ecx
9081      <1>      mov cl, 66/4
9082      <1>      rep movsd
9083      <1>      xor eax, eax
9084      <1>      mov cl, (256-68)/4 ; 188 bytes
9085      <1>      rep stosd
9086      <1>      ; 2 bytes
9087      <1>      stosw
9088      <1>      pop edi
9089      <1>      mov esi, VBE3MODEINFOBLOCK
9090      <1>      ;mov cx, 256
9091      <1>      inc ch ; cx = 256
9092      <1>      call transfer_to_user_buffer
9093      <1>      jnc short vbe_rmi_5
9094      <1>      vbe_rmi_4:
9095      <1>      ;mov eax, 014Fh ; fail/error
9096      <1>      xor eax, eax
9097      <1>      mov ah, 01h
9098      <1>      ;jmp short vbe_rmi_6
9099      <1>      jmp vbe_sm_ret1 ; 11/12/2020

```

```

9099 <1> vbe_rmi_5:
9100 <1> ; 256 bytes of MODEINFO have been transferred to user
9101 <1> ;mov     eax, 4Fh ; succesfull
9102 <1> vbe_rmi_6: ; 12/12/2020
9103 000039A3 31C0 <1> xor     eax, eax
9104 <1> ;vbe_rmi_6:
9105 000039A5 EB7D <1> jmp     vbe_sm_ret1 ; 11/12/2020
9106 <1>
9107 <1> ;pop     edi ; *****
9108 <1> ;pop     ebx ; *****
9109 <1> ;pop     ecx ; *****
9110 <1> ;pop     edx ; *****
9111 <1>
9112 <1> ;;pop    esi ; ****
9113 <1> ;;pop    ebp ; ***
9114 <1> ;;pop    es  ; **
9115 <1> ;;pop    ds  ; *
9116 <1>
9117 <1> ;retn
9118 <1>
9119 <1> set_lfbinfo_table:
9120 <1> ; 19/12/2020
9121 <1> ; 11/12/2020
9122 <1> ; Set/Fill LFBINFO structure/table
9123 <1> ;
9124 <1> ; Input:
9125 <1> ; esi = Mode info list address
9126 <1> ; Output:
9127 <1> ; LFB_Info address is filled with LFBINFO
9128 <1> ; edi = LFB_Info address
9129 <1> ;
9130 <1> ; Modified registers: eax, edx (=0), edi
9131 <1>
9132 000039A7 BF[1A9D0100] <1> mov     edi, LFB_Info
9133 000039AC 8B462A <1> mov     eax, [esi+MODEINFO.PhysBasePtr]
9134 000039AF 894702 <1> mov     [edi+LFBINFO.LFB_addr], eax ; LFB address
9135 <1> ;mov     ax, [esi+MODEINFO.mode]
9136 000039B2 668B06 <1> mov     ax, [esi]
9137 000039B5 668907 <1> mov     [edi+LFBINFO.mode], ax
9138 000039B8 8A461B <1> mov     al, [esi+MODEINFO.BitsPerPixel]
9139 000039BB 88470E <1> mov     [edi+LFBINFO.bpp], al
9140 000039BE 29C0 <1> sub     eax, eax
9141 000039C0 668B4614 <1> mov     ax, [esi+MODEINFO.XResolution]
9142 000039C4 6689470A <1> mov     [edi+LFBINFO.X_res], ax
9143 000039C8 89C2 <1> mov     edx, eax ; 19/12/2020
9144 000039CA 668B4616 <1> mov     ax, [esi+MODEINFO.YResolution]
9145 000039CE 6689470C <1> mov     [edi+LFBINFO.Y_res], ax
9146 <1> ; eax = Y_res ; screen height
9147 <1> ; 19/12/2020
9148 000039D2 F7E2 <1> mul     edx ; X_res*Y_res
9149 <1> ; edx = 0
9150 000039D4 8A570E <1> mov     dl, [edi+LFBINFO.bpp]
9151 <1> ; Note:
9152 <1> ; Bits per pixel may be 8,16,24,32 for TRDOS 386 v2.
9153 <1> ; (4 bits for pixel is not used for VESA modes here)
9154 000039D7 C0EA03 <1> shr     dl, 3 ; convert bits to byte
9155 000039DA F7E2 <1> mul     edx
9156 <1> ; eax = screen/page/buffer size in bytes
9157 000039DC 894706 <1> mov     [edi+LFBINFO.LFB_size], eax
9158 <1> ; edx = 0
9159 <1> ; clear reserved byte in LFBINFO structure/table
9160 000039DF 88570F <1> mov     [edi+LFBINFO.reserved], dl ; not necessary
9161 000039E2 C3 <1> retn
9162 <1>
9163 <1> ; * (TRDOS 386, INT 31h, VESA Video Bios functions)
9164 <1> ; * -----
9165 <1> ; * Function 02h - Set VBE Mode
9166 <1> ; * -----
9167 <1> ; * Input:
9168 <1> ; * AX = 4F02h
9169 <1> ; * BX = Desired Mode to set
9170 <1> ; * Output:
9171 <1> ; * AX = VBE Return Status
9172 <1> ; *
9173 <1> ; * -----
9174 <1> ; *
9175 <1>
9176 <1> vbe_biosfn_set_mode:
9177 <1> ; 07/03/2021
9178 <1> ; 12/12/2020
9179 <1> ; 11/12/2020 (LFBINFO table for VESA VBE modes)
9180 <1> ; 27/11/2020
9181 <1> ; 25/11/2020
9182 <1> ; 23/11/2020 (TRDOS 386 v2.0.3)
9183 <1> ; (ref: vbe.c, 02/01/2020, vruppert)
9184 <1> ;
9185 <1> ; Input:
9186 <1> ; bx = video (bios) mode
9187 <1> ; ax = 4F02h
9188 <1> ; Output:
9189 <1> ; ax = 004Fh (successful)
9190 <1> ; ah > 0 -> error
9191 <1> ;
9192 <1> ; Modified registers: esi
9193 <1>
9194 <1> ; 27/11/2020
9195 <1>
9196 <1> ;;push   ds ; *
9197 <1> ;;push   es ; **
9198 <1> ;;push   ebp ; ***
9199 <1> ;;push   esi ; ****
9200 <1>
9201 <1> ; 11/12/2020
9202 000039E3 52 <1> push    edx ; *****
9203 000039E4 51 <1> push    ecx ; *****
9204 000039E5 53 <1> push    ebx ; *****
9205 000039E6 57 <1> push    edi ; *****
9206 <1>
9207 <1> ;xor     eax, eax
9208 000039E7 80E7C1 <1> and     bh, 0C1h ; use bit 15, 14, 8 only (for bh)
9209 000039EA 883D[0D9D0100] <1> mov     [vbe_mode_x], bh
9210 000039F0 80E701 <1> and     bh, 1
9211 000039F3 753C <1> jnz     short vbe_sm_3 ; VESA VBE mode
9212 <1>
9213 <1> ;;test   bx, 4000h ; VBE_MODE_LINEAR_FRAME_BUFFER
9214 <1> ;test    bh, 40h
9215 <1> ;jz      short vbe_sm_0
9216 <1> ;; lfb_flag
9217 <1> ;mov     al, 40h ; VBE_DISPI_LFB_ENABLED
9218 <1> vbe_sm_0:
9219 <1> ; 27/11/2020
9220 000039F5 B080 <1> mov     al, 80h
9221 <1> ;test    bh, 80h ; VBE_MODE_PRESERVE_DISPLAY_MEMORY
9222 <1> ;jnz     short vbe_sm_1 ; no_clear

```

```

9223 <1> ;; clear
9224 <1> sub al, al ; 0
9225 000039F7 8405[0D9D0100] <1> test [vbe_mode_x], al ; 80h
9226 000039FD 7402 <1> jz short vbe_sm_1 ; clear display memory
9227 <1> ; no_clear
9228 000039FF 08C3 <1> or bl, al ; VBE_MODE_PRESERVE_DISPLAY_MEMORY
9229 <1> vbe_sm_1:
9230 <1> ; check non vesa mode
9231 <1> ;;cmp bx, 100h ; VBE_MODE_VESA_DEFINED
9232 <1> ;;jna short vbe_sm_2
9233 <1> ;and bh, 1
9234 <1> ;jnz short vbe_sm_3
9235 <1>
9236 <1> ; BX <= 1FFh
9237 <1>
9238 <1> ; 27/11/2020
9239 <1> ;or bl, al ; al = 80h if no_clear option is set
9240 <1> ; ; al = 0 if no_clear option is not set
9241 <1>
9242 <1> ; 25/11/2020
9243 <1> ; VBE DISPI will be disabled in 'biosfn_set_video_mode'
9244 <1>
9245 <1> ;xor al, al ; 0 ; VBE_DISPI_DISABLED
9246 <1> ;call dispi_set_enable
9247 <1>
9248 <1> ; call the vgabios in order to set the video mode
9249 <1> ; this allows for going back to textmode with a VBE call
9250 <1> ; (some applications expect that to work)
9251 <1>
9252 <1> ;and bx, 0FFh
9253 <1>
9254 <1> ; 27/11/2020
9255 <1> biosfn_set_video_mode:
9256 <1> ; _call: call subroutine
9257 <1> ; 26/11/2020 (TRDOS 386 v2.0.3)
9258 <1> ; (ref: vgabios.c, 02/01/2020, vruppert)
9259 <1> ; Input:
9260 <1> ; bl = VGA video (bios) mode
9261 <1> ; Output:
9262 <1> ; cf = 1 -> error
9263 <1> ; cf = 0 -> ok
9264 <1> ;
9265 <1> ; Modified registers: esi
9266 <1>
9267 <1> ; 'dispi_set_enable(VBE_DISPI_DISABLED);'
9268 <1>
9269 <1> ;mov ax, 0 ; VBE_DISPI_DISABLED
9270 00003A01 31C0 <1> xor eax, eax ; 0
9271 00003A03 E89C040000 <1> call dispi_set_enable
9272 <1>
9273 00003A08 88D8 <1> mov al, bl
9274 <1> ;jmp _set_mode ; (in 'biosfn_set_video_mode' sub)
9275 00003A0A E8AE1FFFFF <1> call _set_mode ; will return with cf=1 only if
9276 <1> ; desired mode is not implemented
9277 <1> ; _retn: return from subroutine
9278 00003A0F 721A <1> jc short vbe_sm_2 ; 25/11/2020
9279 <1>
9280 <1> ; 26/11/2020
9281 00003A11 31C0 <1> xor eax, eax
9282 00003A13 A01E680000 <1> mov al, [CRT_MODE]
9283 <1> ; 27/11/2020
9284 00003A18 8A25[37830100] <1> mov ah, [noclearmem] ; 80h or 0
9285 <1> ;and ah, 80h
9286 00003A1E 66A3[0E9D0100] <1> mov [video_mode], ax ; bit 15 = no_clear flag
9287 <1> ; bit 14 = 0 (not LFB model)
9288 <1> vbe_sm_ret1:
9289 <1> ; 11/12/2020
9290 <1> ; (vbe_rmi_4 and vbe_rmi_6 jump here)
9291 <1> ; 27/11/2020
9292 00003A24 B04F <1> mov al, 4Fh ; Function call successful
9293 <1> ; eax = 004Fh
9294 <1> vbe_sm_ret2:
9295 <1> ; 11/12/2020
9296 00003A26 5F <1> pop edi ; *****
9297 00003A27 5B <1> pop ebx ; *****
9298 00003A28 59 <1> pop ecx ; *****
9299 00003A29 5A <1> pop edx ; *****
9300 <1>
9301 <1> ;;pop esi ; ****
9302 <1> ;;pop ebp ; ***
9303 <1> ;;pop es ; **
9304 <1> ;;pop ds ; *
9305 <1>
9306 00003A2A C3 <1> retn
9307 <1>
9308 <1> vbe_sm_2:
9309 <1> ;mov ax, 0100h ; Function is not supported
9310 <1> ; 27/11/2020
9311 00003A2B 31C0 <1> xor eax, eax
9312 00003A2D B401 <1> mov ah, 01h
9313 <1> ; eax = 0100h
9314 <1> ;retn
9315 00003A2F EBF5 <1> jmp short vbe_sm_ret2
9316 <1>
9317 <1> vbe_sm_3:
9318 <1> ; 12/12/2020
9319 <1> ; check current mode, if it is 03h
9320 <1> ; save page contents and cursor positions
9321 00003A31 803D[1E680000]03 <1> cmp byte [CRT_MODE], 03h
9322 <1> ;jne short vbe_sm_0
9323 00003A38 7505 <1> jne short vbe_sm_4 ; 07/03/2021
9324 00003A3A E8CE3FFFFF <1> call save_mode3_multiscreen
9325 <1> ; set current mode to extended (SVGA) mode
9326 <1> ;mov byte [CRT_MODE], 0FFh ; VESA VBE mode
9327 <1> vbe_sm_4:
9328 <1> ; 27/11/2020
9329 <1> ; bx = mode (bit 0 to 8)
9330 <1>
9331 <1> ; 25/11/2020
9332 <1>
9333 <1> ; Alternative 2 (instead of 'Mode_info_find_mode')
9334 <1> ;push edi
9335 00003A3F E870050000 <1> call set_mode_info_list ; (alternative 2)
9336 <1> ;pop edi
9337 <1>
9338 <1> ;mov bx, [esi] ; mode
9339 <1>
9340 <1> ; Alternative 1 (instead of 'set_mode_info_list')
9341 <1> ;call mode_info_find_mode ; (alternative 1)
9342 <1>
9343 00003A44 09F6 <1> or esi, esi
9344 00003A46 74E3 <1> jz short vbe_sm_2 ; VBE mode number is wrong
9345 <1> ; or it is not supported
9346 <1>

```

```

9347 <1> ; 11/12/2020
9348 00003A48 668B1E <1> mov bx, [esi] ; mode
9349 <1>
9350 <1> ; 27/11/2020
9351 00003A4B 0A3D[0D9D0100] <1> or bh, [vbe_mode_x]
9352 <1>
9353 <1> ; save VESA VBE mode
9354 00003A51 66891D[0E9D0100] <1> mov [video_mode], bx
9355 <1> ; 27/11/2020
9356 <1> ; bit 0 to 8 = VESA VBE mode
9357 <1> ; bit 9 to 13 = 0 (bit 0 to 13 = mode)
9358 <1> ; bit 14 = Linear/Flat Frame Buffer flag
9359 <1> ; bit 15 = 'memory not cleared
9360 <1> ; at last mode set' flag
9361 <1>
9362 <1> ; first disable current mode
9363 <1> ; (when switching between vesa modes)
9364 <1> ; 'dispi_set_enable(VBE_DISPI_DISABLED);'
9365 <1>
9366 <1> ;mov ax, VBE_DISPI_DISABLED ; 0
9367 00003A58 29C0 <1> sub eax, eax ; 0
9368 <1>
9369 00003A5A E845040000 <1> call dispi_set_enable
9370 <1>
9371 <1> ; 11/12/2020
9372 00003A5F 8A461B <1> mov al, [esi+MODEINFO.BitsPerPixel]
9373 <1> ; ah = 0
9374 <1>
9375 <1> ;cmp byte [esi+MODEINFO.BitsPerPixel], 8
9376 00003A62 3C08 <1> cmp al, 8
9377 00003A64 750B <1> jne short vbe_sm_5
9378 <1>
9379 <1> ; 11/12/2020
9380 <1> ;push edi
9381 00003A66 50 <1> push eax
9382 <1> ; 'load_dac_palette(3);'
9383 00003A67 56 <1> push esi
9384 00003A68 B403 <1> mov ah, 3 ; palette3, 256 colors
9385 00003A6A E8BBF1FFFF <1> call load_dac_palette
9386 00003A6F 5E <1> pop esi
9387 <1> ; 11/12/2020
9388 00003A70 58 <1> pop eax
9389 <1> ;pop edi
9390 <1> vbe_sm_5:
9391 <1> ;'dispi_set_bpp(cur_info->info.BitsPerPixel);'
9392 <1> ; 11/12/2020 (al = bits per pixel, ah = 0)
9393 <1> ;xor ah, ah
9394 <1> ;mov al, [esi+MODEINFO.BitsPerPixel]
9395 00003A71 E841040000 <1> call dispi_set_bpp
9396 <1> ;'dispi_set_xres(cur_info->info.XResolution);'
9397 00003A76 668B4614 <1> mov ax, [esi+MODEINFO.XResolution]
9398 00003A7A E83E040000 <1> call dispi_set_xres
9399 <1> ;'dispi_set_yres(cur_info->info.YResolution);'
9400 00003A7F 668B4616 <1> mov ax, [esi+MODEINFO.YResolution]
9401 00003A83 E83B040000 <1> call dispi_set_yres
9402 <1>
9403 <1> ;'dispi_set_bank(0);'
9404 <1> ;xor ax, ax
9405 00003A88 31C0 <1> xor eax, eax ; 0
9406 00003A8A E83A040000 <1> call dispi_set_bank
9407 <1> ;'dispi_set_enable(VBE_DISPI_ENABLED|no_clear|lfb_flag);'
9408 <1> ;mov ax, di
9409 <1> ; ah = 0 ; 27/11/2020
9410 00003A8F A0[0D9D0100] <1> mov al, [vbe_mode_x] ; restore VBE mode bit 14 & 15
9411 00003A94 0C01 <1> or al, 1 ; VBE_DISPI_ENABLED
9412 00003A96 E809040000 <1> call dispi_set_enable
9413 <1>
9414 <1> ; 'vga_compat_setup();'
9415 00003A9B E83E040000 <1> call vga_compat_setup
9416 <1>
9417 <1> ; 11/12/2020
9418 00003AA0 E802FFFFFF <1> call set_lfbinfo_table
9419 <1>
9420 <1> ; 26/11/2020
9421 00003AA5 31C0 <1> xor eax, eax
9422 00003AA7 FEC8 <1> dec al
9423 00003AA9 A2[1E680000] <1> mov [CRT_MODE], al ; 0FFh = VESA VBE mode sign
9424 <1>
9425 <1> ; 27/11/2020
9426 00003AAE E971FFFFFF <1> jmp vbe_sm_ret1 ; Function call successful
9427 <1>
9428 <1> ; 27/11/2020
9429 <1> ;mov al, 4Fh
9430 <1> ; ; eax = 004Fh = Function call successful
9431 <1> ;jmp short vbe_sm_ret2
9432 <1>
9433 <1> ; * (TRDOS 386, INT 31h, VESA Video Bios functions)
9434 <1> ; * -----
9435 <1> ; * Function 03h - Return Current VBE Mode
9436 <1> ; * -----
9437 <1> ; * Input:
9438 <1> ; * AX = 4F03h
9439 <1> ; * Output:
9440 <1> ; * AX = VBE Return Status
9441 <1> ; * BX = Current VBE Mode
9442 <1> ; * -----
9443 <1> ; * -----
9444 <1> ; *
9445 <1> ;
9446 <1> vbe_biosfn_return_current_mode:
9447 <1> ; 11/12/2020
9448 <1> ; 27/11/2020 (TRDOS 386 v2.0.3)
9449 <1> ; (ref: vbe.c, 02/01/2020, vruppert)
9450 <1> ;
9451 <1> ; Input:
9452 <1> ; none
9453 <1> ; Output:
9454 <1> ; ax = 004Fh (successful)
9455 <1> ; ah > 0 -> error
9456 <1> ; bx = current video (bios) mode (if ah = 0)
9457 <1> ;
9458 <1> ; Modified registers: eax, ebx
9459 <1>
9460 <1> ; 27/11/2020
9461 <1>
9462 <1> ;;push ds ; *
9463 <1> ;;push es ; **
9464 <1> ;;push ebp ; ***
9465 <1> ;;push esi ; ****
9466 <1>
9467 <1> ;push edx ; *****
9468 <1>
9469 <1> ; (vbe.c)
9470 <1> ;call dispi_get_enable

```

```

9471      <1>      ;      ; ax = vbe display interface status
9472      <1>      ;and    al, 1 ; VBE_DISPI_ENABLED
9473      <1>      ;jnz    short vbe_gm_1 ; VBE graphics mode
9474      <1>
9475 00003AB3 A0[1E680000] <1>      mov    al, [CRT_MODE] ; current cga/vga mode
9476 00003AB8 3CFF      <1>      cmp    al, 0FFh ; VBE extension signature
9477 00003ABA 720E      <1>      jb     short vbe_gm_1 ; get CGA/VGA mode
9478      <1>
9479      <1>      ; get VBE mode
9480      <1> vbe_gm_0:
9481 00003ABC 66A1[0E9D0100] <1>      mov    ax, [video_mode]
9482      <1>      ; BX bits:
9483      <1>      ; bit 0 to 8 = VESA VBE video mode
9484      <1>      ; bit 9 to 13 = 0
9485      <1>      ; bit 14 = last mode set LFB option
9486      <1>      ;          1 - linear/flat frame buffer
9487      <1>      ;          0 - windowed frame buffer
9488      <1>      ; bit 15 = last mode set no_clear option
9489      <1>      ;          0 - video memory cleared
9490      <1>      ;          1 - video memory not cleared
9491      <1>
9492      <1> vbe_gm_return:
9493      <1>      ;pop     edx ; *****
9494 00003AC2 0FB7D8      <1>      movzx  ebx, ax
9495      <1> ;vbe_srs_retn:
9496 00003AC5 31C0      <1>      xor     eax, eax ; 0
9497 00003AC7 B04F      <1>      mov    al, 4Fh ; ax = 004Fh (successful)
9498 00003AC9 C3      <1>      retn
9499      <1>
9500      <1> vbe_gm_1:
9501      <1>      ; legacy (old, standard) CGA/VGA bios video mode
9502 00003ACA 8A25[37830100] <1>      mov    ah, [noclearmem] ; 80h or 0
9503      <1>      ; BX bits:
9504      <1>      ; bit 0 to 7 = video mode
9505      <1>      ; bit 8 to 13 = 0
9506      <1>      ; bit 14 = 0 (not LFB mode) CGA/VGA
9507      <1>      ; bit 15 = 1 if [noclearmem] = 80h
9508      <1>      ;          0 if [noclearmem] = 0
9509 00003AD0 EBF0      <1>      jmp     short vbe_gm_return
9510      <1>
9511      <1> ; * (TRDOS 386, INT 31h, VESA Video Bios functions)
9512      <1> ; * -----
9513      <1> ; * Function 04h - Save/Restore State
9514      <1> ; * -----
9515      <1> ; * Input:
9516      <1> ; *      AX = 4F04h
9517      <1> ; *      DL = 00h Return Save/Restore State buff size
9518      <1> ; *      01h Save State
9519      <1> ; *      02h Restore State
9520      <1> ; *      CX = Requested states
9521      <1> ; *      bit 0 - controller hardware state
9522      <1> ; *      bit 1 - BIOS data state
9523      <1> ; *      bit 2 - DAC state
9524      <1> ; *      bit 3 - register state
9525      <1> ; *      (ES:BX) EBX = Pointer to buffer (if DL <> 00h)
9526      <1> ; * Output:
9527      <1> ; *      AX = VBE Return Status
9528      <1> ; *      BX = Number of 64-byte blocks
9529      <1> ; *      to hold the state buffer (if DL=00h)
9530      <1> ; * -----
9531      <1> ; *
9532      <1> ; *
9533      <1> ; *
9534      <1> vbe_biosfn_save_restore_state:
9535      <1>      ; 23/01/2021
9536      <1>      ; 16/01/2021
9537      <1>      ; 14/01/2021
9538      <1>      ; 13/01/2021
9539      <1>      ; 12/01/2021
9540      <1>      ; 11/01/2021 (TRDOS 386 v2.0.3)
9541      <1>      ; (ref: vbe.c, 02/01/2020, vruppert)
9542      <1>
9543      <1>      ; Input:
9544      <1>      ;      dl = sub function
9545      <1>      ;      cl = requested state
9546      <1>      ;      ebx = pointer to buffer (if dl<>00h)
9547      <1>      ; Output:
9548      <1>      ;      ax = 004Fh (successful)
9549      <1>      ;      ah > 0 -> error
9550      <1>      ;      bx = Number of 64-byte blocks
9551      <1>      ;      to hold the state buffer (if DL=00h)
9552      <1>
9553      <1>      ; Modified registers: eax, ebx, edi
9554      <1>
9555      <1>      ; 14/01/2021
9556 00003AD2 09DB      <1>      or     ebx, ebx ; user's buffer address
9557 00003AD4 750A      <1>      jnz     short _vbe_biosfn_save_restore_state
9558      <1>
9559 00003AD6 20D2      <1>      and     dl, dl
9560 00003AD8 7406      <1>      jz      short _vbe_biosfn_save_restore_state
9561      <1>
9562      <1>      ; function failed
9563      <1>      ;mov     eax, 0100h
9564      <1>      ;xor     eax, eax
9565      <1>      ;inc     ah ; eax = 0100h
9566      <1>      ; 16/01/2021
9567 00003ADA B84F010000 <1>      mov     eax, 014Fh
9568 00003ADF C3      <1>      retn
9569      <1>
9570      <1> _vbe_biosfn_save_restore_state:
9571      <1>      ; 23/01/2021
9572      <1>      ; 14/01/2021
9573      <1>      ; ebx = 0 if the caller is kernel ('sysvideo')
9574      <1>
9575      <1>      ; 13/01/2021
9576 00003AE0 57      <1>      push    edi
9577 00003AE1 52      <1>      push    edx
9578 00003AE2 51      <1>      push    ecx
9579      <1>
9580      <1>      ; 23/01/2021
9581      <1>      ; 12/01/2021
9582 00003AE3 80FA02 <1>      cmp     dl, 2
9583 00003AE6 7757 <1>      ja     short vbe_srs_7 ; 23/01/2021
9584      <1>      ; invalid sub function
9585 00003AE8 83F90F <1>      cmp     ecx, 0Fh
9586 00003AEB 7752 <1>      ja     short vbe_srs_7 ; invalid !
9587      <1>
9588 00003AED 20D2 <1>      and     dl, dl
9589 00003AEF 7515 <1>      jnz     short vbe_srs_4
9590      <1>
9591      <1>      ; DL = 0
9592      <1>      ; Return Save/Restore State buffer size
9593      <1>
9594      <1>      ;mov     ebx, ecx

```

```

9595     <1>      ;shl     bl, 1
9596     <1>      ;mov     bx, [ebx+vbestatebufsize]
9597 00003AF1 E881000000 <1>      call    vbe_srs_gbs
9598     <1>
9599     <1>      ; 11/01/2021
9600     <1>      ; test    cl, 8
9601     <1>      ; jz      short vbe_srs_3
9602     <1>      ; ; vbe_biosfn_read_video_state_size();
9603     <1>      ; ; return 9 * 2;
9604     <1>      ; mov     bl, 18 ; register state size
9605     <1> ;vbe_srs_0:
9606     <1>      ; test    cl, 1
9607     <1>      ; jz      short vbe_srs_1
9608     <1>      ; ; size += 0x46;
9609     <1>      ; add     bl, 70 ; controller state size
9610     <1> ;vbe_srs_1:
9611     <1>      ; test    cl, 2
9612     <1>      ; jz      short vbe_srs_2
9613     <1>      ; ; size += (5 + 8 + 5) * 2 + 6;
9614     <1>      ; add     bl, 42 ; BIOS data state size ; Bochs/Plex86
9615     <1>      ; ; 12/01/2021
9616     <1>      ; add     bl, 40 ; TRDOS 386 v2 VBIOS data state size
9617     <1> ;vbe_srs_2:
9618     <1>      ; test    cl, 4
9619     <1>      ; jz      short vbe_srs_3
9620     <1>      ; ; size += 3 + 256 * 3 + 1;
9621     <1>      ; add     bx, 772 ; DAC state size
9622     <1>
9623     <1> vbe_srs_3:
9624 00003AF6 6683C33F <1>      add     bx, 63
9625 00003AFA 66C1EB06 <1>      shr     bx, 6 ; / 64
9626     <1>
9627     <1> vbe_srs_retn:
9628 00003AFE 31C0 <1>      xor     eax, eax ; 0
9629     <1> vbe_srs_0: ; 16/01/2021
9630 00003B00 B04F <1>      mov     al, 4Fh ; ax = 004Fh (successful)
9631     <1> ;vbe_srs_0:
9632     <1>      ; 13/01/2021
9633 00003B02 59 <1>      pop     ecx
9634 00003B03 5A <1>      pop     edx
9635 00003B04 5F <1>      pop     edi
9636     <1>
9637 00003B05 C3 <1>      retn
9638     <1>
9639     <1>      ; 23/01/2021
9640     <1> ;vbe_srs_10:
9641     <1>      ; ; 14/01/2021
9642     <1>      ; ; return to 'sysvideo'
9643     <1>      ; mov     ebx, ecx ; transfer count
9644     <1>      ; ; (byte count for saving current video state)
9645     <1>      ; jmp     short vbe_srs_retn
9646     <1>
9647     <1> vbe_srs_4:
9648     <1>      ; 23/01/2021
9649 00003B06 80E10F <1>      and     cl, 0Fh ; 8, 4, 2, 1
9650 00003B09 7434 <1>      jz      short vbe_srs_7 ; cx = 0 -> invalid !
9651     <1>
9652 00003B0B BF00760900 <1>      mov     edi, VBE3SAVERESTOREBLOCK
9653     <1>
9654 00003B10 80FA01 <1>      cmp     dl, 1
9655 00003B13 7730 <1>      ja      short vbe_srs_8
9656     <1>
9657     <1>      ; save video state
9658     <1>
9659 00003B15 F6C107 <1>      test    cl, 07h ; 4, 2, 1
9660 00003B18 740A <1>      jz      short vbe_srs_5 ; vbe dispi regs state
9661     <1>
9662 00003B1A E884000000 <1>      call    biosfn_save_video_state
9663     <1>      ; edi = current position
9664     <1>      ; in VBE3SAVERESTOREBLOCK
9665     <1>      ; (VGA save_state offset)
9666     <1>      ; modified regs: edi, eax, edx, ch
9667 00003B1F F6C108 <1>      test    cl, 8
9668 00003B22 7405 <1>      jz      short vbe_srs_6
9669     <1> vbe_srs_5:
9670 00003B24 E8AB010000 <1>      call    vbe_biosfn_save_video_state
9671     <1>      ; edi = end position
9672     <1>      ; in VBE3SAVERESTOREBLOCK
9673     <1>      ; (VGA save_state offset)
9674     <1>      ; modified regs: edi, eax, edx, ch
9675     <1> vbe_srs_6:
9676     <1>      ; 23/01/2021
9677 00003B29 21DB <1>      and     ebx, ebx
9678 00003B2B 74D1 <1>      jz      short vbe_srs_retn ; the caller is kernel
9679     <1>
9680 00003B2D BE00760900 <1>      mov     esi, VBE3SAVERESTOREBLOCK
9681 00003B32 29F7 <1>      sub     edi, esi
9682 00003B34 89F9 <1>      mov     ecx, edi ; transfer count in bytes
9683     <1>
9684     <1>      ; ; 14/01/2021
9685     <1>      ; and     ebx, ebx
9686     <1>      ; jz      short vbe_srs_10 ; the caller is kernel
9687     <1>
9688 00003B36 89DF <1>      mov     edi, ebx ; user's buffer address
9689 00003B38 E8CECF0000 <1>      call    transfer_to_user_buffer
9690 00003B3D 73BF <1>      jnc     short vbe_srs_retn
9691     <1> vbe_srs_7:
9692     <1>      ; // function failed
9693     <1>      ; mov     eax, 0100h
9694 00003B3F 31C0 <1>      xor     eax, eax
9695 00003B41 FEC4 <1>      inc     ah ; eax = 0100h
9696     <1>      ; 16/01/2021
9697     <1>      ; ax = 0014Fh
9698     <1>      ; retn
9699     <1>      ; 13/01/2021
9700 00003B43 EBBB <1>      jmp     short vbe_srs_0
9701     <1> vbe_srs_8:
9702     <1>      ; cmp     dl, 2
9703     <1>      ; jne     short vbe_srs_7
9704     <1>      ; ; invalid sub function
9705     <1>
9706     <1>      ; 14/01/2021
9707 00003B45 09DB <1>      or      ebx, ebx ; user's buffer address
9708     <1>      ; jnz     short vbe_srs_11
9709     <1>
9710     <1>      ; the caller is kernel ('sysvideo')
9711     <1>      ; jmp     short vbe_srs_12
9712     <1>      ; 23/01/2021
9713 00003B47 7414 <1>      jz      short vbe_srs_12 ; 'sysvideo' call
9714     <1> vbe_srs_11:
9715 00003B49 89DE <1>      mov     esi, ebx ; user's buffer address
9716     <1>      ; 23/01/2021
9717     <1>      ; push    ebx
9718     <1>

```



```

9719 00003B4B E827000000 <1> call vbe_srs_gbs
9720 <1>
9721 <1> ; restore video state
9722 <1>
9723 <1> ;mov edi, VBE3SAVERESTOREBLOCK
9724 00003B50 51 <1> push ecx
9725 00003B51 89D9 <1> mov ecx, ebx ; transfer count in bytes
9726 00003B53 E8FDCF0000 <1> call transfer_from_user_buffer
9727 00003B58 59 <1> pop ecx
9728 <1> ; 23/01/2021
9729 <1> ;pop ebx
9730 00003B59 89F3 <1> mov ebx, esi
9731 00003B5B 72E2 <1> jc short vbe_srs_7
9732 <1>
9733 <1> vbe_srs_12:
9734 <1> ;mov esi, VBE3SAVERESTOREBLOCK
9735 00003B5D 89FE <1> mov esi, edi
9736 <1>
9737 00003B5F F6C107 <1> test cl, 07h ; 4, 2, 1
9738 00003B62 740C <1> jz short vbe_srs_9 ; vbe dispi regs state
9739 <1>
9740 00003B64 E8A2010000 <1> call biosfn_restore_video_state
9741 00003B69 72D4 <1> jc short vbe_srs_7 ; invalid buffer content !
9742 <1> ; esi = current position
9743 <1> ; in VBE3SAVERESTOREBLOCK
9744 <1> ; (VGA save_state offset)
9745 <1> ; modified regs: esi, eax, edx, ch
9746 00003B6B F6C108 <1> test cl, 8
9747 <1> ;jz short vbe_srs_10
9748 <1> ; 23/01/2020
9749 00003B6E EB8E <1> jmp short vbe_srs_retn
9750 <1> vbe_srs_9:
9751 00003B70 E8F1020000 <1> call vbe_biosfn_restore_video_state
9752 <1>
9753 <1> ; modified regs: esi, eax, edx, ch
9754 <1>
9755 00003B75 EB87 <1> jmp short vbe_srs_retn
9756 <1>
9757 <1> ;vbe_srs_10:
9758 <1> ; ; successful
9759 <1> ; xor eax, eax ; 0
9760 <1> ; mov al, 4Fh ; ax = 004Fh (successful)
9761 <1> ; retn
9762 <1>
9763 <1> vbe_srs_gbs:
9764 <1> ; return buffer size according to flags
9765 00003B77 89CB <1> mov ebx, ecx ; options/flags
9766 00003B79 D0E3 <1> shl bl, 1
9767 00003B7B 668B9B[833B0000] <1> mov bx, [ebx+vbestatebufsize]
9768 00003B82 C3 <1> retn
9769 <1>
9770 <1> vbestatebufsize:
9771 <1> ; -----
9772 <1> ; CL = 0 1 2 3 4 5 6 7
9773 <1> ; -----
9774 00003B83 0000460028006E0004- <1> dw 0, 70, 40, 110, 772, 842, 812, 882
9774 00003B8C 034A032C037203 <1>
9775 <1> ; -----
9776 <1> ; CL = 8 9 10 11 12 13 14 15
9777 <1> ; -----
9778 00003B93 120058003A00800016- <1> dw 18, 88, 58, 128, 790, 860, 830, 900
9778 00003B9C 035C033E038403 <1>
9779 <1>
9780 <1> ; 11/01/2021
9781 <1> VGAREG_ACTL_ADDRESS equ 3C0h
9782 <1> VGAREG_ACTL_WRITE_DATA equ 3C0h
9783 <1> VGAREG_ACTL_READ_DATA equ 3C1h
9784 <1>
9785 <1> VGAREG_INPUT_STATUS equ 3C2h
9786 <1> VGAREG_WRITE_MISC_OUTPUT equ 3C2h
9787 <1> VGAREG_VIDEO_ENABLE equ 3C3h
9788 <1> VGAREG_SEQU_ADDRESS equ 3C4h
9789 <1> VGAREG_SEQU_DATA equ 3C5h
9790 <1>
9791 <1> VGAREG_PEL_MASK equ 3C6h
9792 <1> VGAREG_DAC_STATE equ 3C7h
9793 <1> VGAREG_DAC_READ_ADDRESS equ 3C7h
9794 <1> VGAREG_DAC_WRITE_ADDRESS equ 3C8h
9795 <1> VGAREG_DAC_DATA equ 3C9h
9796 <1>
9797 <1> VGAREG_READ_FEATURE_CTL equ 3CAh
9798 <1> VGAREG_READ_MISC_OUTPUT equ 3CCh
9799 <1>
9800 <1> VGAREG_GRDC_ADDRESS equ 3CEh
9801 <1> VGAREG_GRDC_DATA equ 3CFh
9802 <1>
9803 <1> ;VGAREG_MDA_CRTC_ADDRESS equ 3B4h
9804 <1> ;VGAREG_MDA_CRTC_DATA equ 3B5h
9805 <1> VGAREG_VGA_CRTC_ADDRESS equ 3D4h
9806 <1> VGAREG_VGA_CRTC_DATA equ 3D5h
9807 <1>
9808 <1> ;VGAREG_MDA_WRITE_FEATURE_CTL equ 3BAh
9809 <1> VGAREG_VGA_WRITE_FEATURE_CTL equ 3DAh
9810 <1> VGAREG_ACTL_RESET equ 3DAh
9811 <1>
9812 <1> ;VGAREG_MDA_MODECTL equ 3B8h
9813 <1> VGAREG_CGA_MODECTL equ 3D8h
9814 <1> VGAREG_CGA_PALETTE equ 3D9h
9815 <1>
9816 <1> biosfn_save_video_state:
9817 <1> ; 03/08/2022 (TRDOS 386 v2.0.5)
9818 <1> ; 22/01/2021
9819 <1> ; 12/01/2021
9820 <1> ; 11/01/2021 (TRDOS 386 v2.0.3)
9821 <1> ; (vgabios.c)
9822 <1>
9823 <1> ; modified registers: eax, edx, edi, ch
9824 <1>
9825 <1> ;mov edi, VBE3SAVERESTOREBLOCK
9826 <1>
9827 <1> ; input: edi = state buffer address
9828 <1>
9829 00003BA3 F6C101 <1> test cl, 1
9830 00003BA6 0F8485000000 <1> jz bfn_svs_4
9831 <1>
9832 00003BAC 66BAC403 <1> mov dx, VGAREG_SEQU_ADDRESS ; 3C7h
9833 00003BB0 EC <1> in al, dx
9834 00003BB1 AA <1> stosb
9835 <1> ;mov dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9836 00003BB2 B2D4 <1> mov dl, 0D4h
9837 00003BB4 EC <1> in al, dx
9838 00003BB5 AA <1> stosb
9839 <1> ;mov dx, VGAREG_GRDC_ADDRESS ; 3CEh
9840 00003BB6 B2CE <1> mov dl, 0CEh

```

```

9841 00003BB8 EC      <1>      in      al, dx
9842 00003BB9 AA      <1>      stosb
9843                <1>      ;mov     dx, VGAREG_ACTL_RESET ; 3DAh
9844 00003BBA B2DA     <1>      mov     dl, 0DAh
9845 00003BBC EC      <1>      in      al, dx
9846                <1>      ;mov     dx, VGAREG_ACTL_ADDRESS ; 3C0h
9847 00003BBD B2C0     <1>      mov     dl, 0C0h
9848 00003BBF EC      <1>      in      al, dx
9849 00003BC0 AA      <1>      stosb
9850 00003BC1 88C4     <1>      mov     ah, al ; ar_index
9851                <1>      ;mov     dx, VGAREG_READ_FEATURE_CTL ; 3CAh
9852 00003BC3 B2CA     <1>      mov     dl, 0CAh
9853 00003BC5 EC      <1>      in      al, dx
9854 00003BC6 AA      <1>      stosb
9855                <1>      ; (5 bytes are written above)
9856                <1>
9857                <1>      ; for(i=1;i<=4;i++){
9858 00003BC7 B001     <1>      mov     al, 1
9859                <1>      ;mov     dx, VGAREG_SEQU_ADDRESS ; 3C4h
9860                <1>      ;mov     dl, 0C4h
9861 00003BC9 B504     <1>      mov     ch, 4
9862                <1>      bfn_svs_0:
9863                <1>      ; outb(VGAREG_SEQU_ADDRESS, i);
9864                <1>      ;mov     dx, VGAREG_SEQU_ADDRESS ; 3C4h
9865 00003BCB B2C4     <1>      mov     dl, 0C4h
9866 00003BCD EE      <1>      out     dx, al
9867                <1>      ;mov     dx, VGAREG_SEQU_DATA ; 3C5h
9868 00003BCE FEC2     <1>      inc     dl ; dx = 3C5h
9869                <1>      ; inb(VGAREG_SEQU_DATA)
9870 00003BD0 50       <1>      push    eax
9871 00003BD1 EC      <1>      in      al, dx
9872 00003BD2 AA      <1>      stosb ; (4 bytes in loop)
9873 00003BD3 58       <1>      pop     eax
9874                <1>      ;mov     dx, VGAREG_SEQU_ADDRESS ; 3C4h
9875                <1>      ;dec     dl
9876 00003BD4 FEC0     <1>      inc     al ; i++
9877 00003BD6 FECD     <1>      dec     ch
9878 00003BD8 75F1     <1>      jnz     short bfn_svs_0
9879                <1>
9880                <1>      ; outb(VGAREG_SEQU_ADDRESS, 0);
9881 00003BDA 28C0     <1>      sub     al, al ; 0
9882 00003BDC EE      <1>      out     dx, al
9883                <1>      ; inb(VGAREG_SEQU_DATA)
9884                <1>      ;mov     dx, VGAREG_SEQU_DATA ; 3C5h
9885 00003BDD FEC2     <1>      inc     dl ; dx = 3C5h
9886 00003BDF EC      <1>      in      al, dx
9887 00003BE0 AA      <1>      stosb ; (+1 byte)
9888                <1>
9889                <1>      ; for(i=0;i<=0x18;i++) {
9890 00003BE1 28C0     <1>      sub     al, al ; 0
9891                <1>      ;mov     dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9892                <1>      ;mov     dl, 0D4h
9893 00003BE3 B519     <1>      mov     ch, 25
9894                <1>      bfn_svs_1:
9895                <1>      ; outb(crtc_addr,i);
9896                <1>      ;mov     dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9897 00003BE5 B2D4     <1>      mov     dl, 0D4h
9898 00003BE7 EE      <1>      out     dx, al
9899                <1>      ;mov     dx, VGAREG_VGA_CRTC_DATA ; 3D5h
9900 00003BE8 FEC2     <1>      inc     dl ; dx = 3D5h
9901                <1>      ; inb(crtc_addr+1)
9902 00003BEA 50       <1>      push    eax
9903 00003BEB EC      <1>      in      al, dx
9904 00003BEC AA      <1>      stosb ; (25 bytes in loop)
9905 00003BED 58       <1>      pop     eax
9906                <1>      ;mov     dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
9907                <1>      ;dec     dl
9908 00003BEE FEC0     <1>      inc     al ; i++
9909 00003BF0 FECD     <1>      dec     ch
9910 00003BF2 75F1     <1>      jnz     short bfn_svs_1
9911                <1>
9912 00003BF4 80E420    <1>      and     ah, 20h ; (ar_index & 0x20)
9913                <1>      ; for(i=0;i<=0x13;i++) {
9914 00003BF7 28C0     <1>      sub     al, al ; 0
9915 00003BF9 B514     <1>      mov     ch, 20
9916                <1>      bfn_svs_2:
9917                <1>      ; inb(VGAREG_ACTL_RESET);
9918                <1>      ;mov     dx, VGAREG_ACTL_RESET ; 3DAh
9919 00003BFB B2DA     <1>      mov     dl, 0DAh
9920 00003BFD 50       <1>      push    eax
9921 00003BFE EC      <1>      in      al, dx
9922 00003BFF 8A0424    <1>      mov     al, [esp]
9923                <1>      ; outb(VGAREG_ACTL_ADDRESS, i | (ar_index & 0x20));
9924 00003C02 08E0     <1>      or      al, ah
9925                <1>      ;mov     dx, VGAREG_ACTL_ADDRESS ; 3C0h
9926 00003C04 B2C0     <1>      mov     dl, 0C0h
9927 00003C06 EE      <1>      out     dx, al
9928                <1>      ;mov     dx, VGAREG_ACTL_READ_DATA ; 3C1h
9929                <1>      ;mov     dl, 0C1h
9930 00003C07 FEC2     <1>      inc     dl
9931 00003C09 EC      <1>      in      al, dx
9932 00003C0A AA      <1>      stosb ; (20 bytes in loop)
9933 00003C0B 58       <1>      pop     eax
9934 00003C0C FEC0     <1>      inc     al ; i++
9935 00003C0E FECD     <1>      dec     ch
9936 00003C10 75E9     <1>      jnz     short bfn_svs_2
9937                <1>
9938                <1>      ; inb(VGAREG_ACTL_RESET);
9939                <1>      ;mov     dx, VGAREG_ACTL_RESET ; 3DAh
9940 00003C12 B2DA     <1>      mov     dl, 0DAh
9941 00003C14 EC      <1>      in      al, dx
9942                <1>
9943                <1>      ; for(i=0;i<=8;i++) {
9944 00003C15 28C0     <1>      sub     al, al ; 0
9945                <1>      ;mov     dx, VGAREG_GRDC_ADDRESS ; 3CEh
9946                <1>      ;mov     dl, 0CEh
9947 00003C17 B509     <1>      mov     ch, 9
9948                <1>      bfn_svs_3:
9949                <1>      ; outb(VGAREG_GRDC_ADDRESS,i)
9950                <1>      ;mov     dx, VGAREG_GRDC_ADDRESS ; 3CEh
9951 00003C19 B2CE     <1>      mov     dl, 0CEh
9952 00003C1B EE      <1>      out     dx, al
9953                <1>      ; inb(VGAREG_ACTL_READ_DATA)
9954 00003C1C 50       <1>      push    eax
9955                <1>      ;mov     dx, VGAREG_GRDC_DATA ; 3CFh
9956                <1>      ;mov     dl, 0CFh
9957 00003C1D FEC2     <1>      inc     dl
9958 00003C1F EC      <1>      in      al, dx
9959 00003C20 AA      <1>      stosb ; (9 bytes in loop)
9960 00003C21 58       <1>      pop     eax
9961                <1>      ;dec     dl
9962 00003C22 FEC0     <1>      inc     al ; i++
9963 00003C24 FECD     <1>      dec     ch
9964 00003C26 75F1     <1>      jnz     short bfn_svs_3

```

```

9965 <1>
9966 <1> ; write_word(ES, BX, crtc_addr); BX+= 2;
9967 <1> ; (offset 64)
9968 00003C28 66B8D403 <1> mov ax, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
9969 00003C2C 66AB <1> stosw ; (2 bytes (1 word))
9970 <1>
9971 <1> ; /* xxx: read plane latches */
9972 00003C2E 31C0 <1> xor eax, eax ; 0
9973 00003C30 AB <1> stosd ; (4 bytes)
9974 <1>
9975 <1> ; (total 70 bytes are written above as controller hardware state)
9976 <1>
9977 <1> bfn_svs_4:
9978 <1> ; 12/01/2021 (TRDOS 386 v2.0.3)
9979 00003C31 F6C102 <1> test cl, 2
9980 00003C34 7476 <1> jz short bfn_svs_6
9981 <1>
9982 <1> ; VIDEO BIOS DATA
9983 <1> ; !!! this data is valid for TRDOS 386 v2 kernel only !!!
9984 <1> ; (this is not same with BOCHS/PLEX86 video bios, BIOS data)
9985 <1>
9986 <1> ; if (CX & 2) {
9987 <1> ; write_byte(ES, BX, read_byte(BIOSMEM_SEG, BIOSMEM_CURRENT_MODE)); BX++;
9988 <1> ; write_word(ES, BX, read_word(BIOSMEM_SEG, BIOSMEM_NB_COLS)); BX += 2;
9989 <1> ; write_word(ES, BX, read_word(BIOSMEM_SEG, BIOSMEM_PAGE_SIZE)); BX += 2;
9990 <1> ; write_word(ES, BX, read_word(BIOSMEM_SEG, BIOSMEM_CRTC_ADDRESS)); BX += 2;
9991 <1> ; write_byte(ES, BX, read_byte(BIOSMEM_SEG, BIOSMEM_NB_ROWS)); BX++;
9992 <1> ; write_word(ES, BX, read_word(BIOSMEM_SEG, BIOSMEM_CHAR_HEIGHT)); BX += 2;
9993 <1> ; write_byte(ES, BX, read_byte(BIOSMEM_SEG, BIOSMEM_VIDEO_CTL)); BX++;
9994 <1> ; write_byte(ES, BX, read_byte(BIOSMEM_SEG, BIOSMEM_SWITCHES)); BX++;
9995 <1> ; write_byte(ES, BX, read_byte(BIOSMEM_SEG, BIOSMEM_MODESET_CTL)); BX++;
9996 <1> ; write_word(ES, BX, read_word(BIOSMEM_SEG, BIOSMEM_CURSOR_TYPE)); BX += 2;
9997 <1> ; for(i=0; i<8; i++) {
9998 <1> ; write_word(ES, BX, read_word(BIOSMEM_SEG, BIOSMEM_CURSOR_POS+2*i));
9999 <1> ; BX += 2;
10000 <1> ; }
10001 <1> ; write_word(ES, BX, read_word(BIOSMEM_SEG, BIOSMEM_CURRENT_START)); BX += 2;
10002 <1> ; write_byte(ES, BX, read_byte(BIOSMEM_SEG, BIOSMEM_CURRENT_PAGE)); BX++;
10003 <1> ; /* current font */
10004 <1> ; write_word(ES, BX, read_word(0, 0x1f * 4)); BX += 2;
10005 <1> ; write_word(ES, BX, read_word(0, 0x1f * 4 + 2)); BX += 2;
10006 <1> ; write_word(ES, BX, read_word(0, 0x43 * 4)); BX += 2;
10007 <1> ; write_word(ES, BX, read_word(0, 0x43 * 4 + 2)); BX += 2;
10008 <1>
10009 <1> ; !!! save TRDOS 386 v2 kernel specific video bios data !!!
10010 <1> ; (which is/are used by 'SET_MODE' function and/or it's sub functions)
10011 <1>
10012 00003C36 66B8D403 <1> mov ax, 3D4h ; CRTC_ADDR, always 3D4h (color VGA) for TRDOS 386 v2
10013 00003C3A 66AB <1> stosw
10014 00003C3C A0[1E680000] <1> mov al, [CRT_MODE] ; Current video mode (0FFh for VESA VBE modes)
10015 00003C41 AA <1> stosb
10016 00003C42 A0[1F680000] <1> mov al, [CRT_MODE_SET] ; 29h for mode 03h ; TRDOS 386 feature only !
10017 00003C47 AA <1> stosb
10018 00003C48 66A1[0E9D0100] <1> mov ax, [video_mode] ; Current VESA VBE (SVGA, extended VGA) mode
10019 00003C4E 66AB <1> stosw ; (valid if [CRT_MODE] = 0FFh)
10020 00003C50 66A1[38830100] <1> mov ax, [CRT_LEN] ; page size (in bytes)
10021 00003C56 66AB <1> stosw
10022 00003C58 66A1[BC760100] <1> mov ax, [CRT_START] ; video page start offset
10023 00003C5E 66AB <1> stosw
10024 00003C60 A0[20680000] <1> mov al, [CRT_COLS] ; nbcols, characters per row
10025 00003C65 AA <1> stosb
10026 00003C66 A0[26680000] <1> mov al, [VGA_ROWS] ; nbrows, (character) rows per page (not rows-1)
10027 00003C6B AA <1> stosb
10028 00003C6C A0[22680000] <1> mov al, [CHAR_HEIGHT] ; character font height (8 or 16 or 14)
10029 00003C71 AA <1> stosb
10030 00003C72 A0[23680000] <1> mov al, [VGA_VIDEO_CTL] ; ROM BIOS DATA AREA Offset 87h
10031 00003C77 AA <1> stosb
10032 00003C78 A0[24680000] <1> mov al, [VGA_SWITCHES] ; feature bit switches
10033 00003C7D AA <1> stosb
10034 00003C7E A0[25680000] <1> mov al, [VGA_MODESET_CTL] ; basic mode set options
10035 00003C83 AA <1> stosb
10036 <1> ; followings are only used by TRDOS 386 v2 (IBM PC/AT ROMBIOS) code
10037 <1> ; (bochs/plex86 does not use and return those)
10038 00003C84 A0[21680000] <1> mov al, [CRT_PALETTE] ; current color palette ; TRDOS 386 feature only !
10039 00003C89 AA <1> stosb
10040 00003C8A A0[CE760100] <1> mov al, [ACTIVE_PAGE] ; current video page
10041 00003C8F AA <1> stosb
10042 00003C90 66A1[37680000] <1> mov ax, [CURSOR_MODE] ; cursor type
10043 00003C96 66AB <1> stosw
10044 <1> ; mov eax, [CURSOR_POSN] ; cursor position for video page 0 and 1
10045 <1> ; stosd
10046 <1> ; mov eax, [CURSOR_POSN+4] ; cursor position for video page 2 and 3
10047 <1> ; stosd
10048 <1> ; mov eax, [CURSOR_POSN+8] ; cursor position for video page 4 and 5
10049 <1> ; stosd
10050 <1> ; mov eax, [CURSOR_POSN+12] ; cursor position for video page 6 and 7
10051 <1> ; stosd
10052 00003C98 56 <1> push esi
10053 00003C99 B504 <1> mov ch, 4
10054 00003C9B BE[BE760100] <1> mov esi, CURSOR_POSN
10055 <1> bfn_svs_5:
10056 00003CA0 A5 <1> movsd
10057 00003CA1 FECD <1> dec ch
10058 00003CA3 75FB <1> jnz short bfn_svs_5
10059 00003CA5 5E <1> pop esi
10060 <1> ; (font addr) protected mode address in kernel's/system memory space
10061 <1> ; (not accessible/meaningful address value by user)
10062 00003CA6 A1[4A830100] <1> mov eax, [VGA_INT43H] ; VGA current (default) font address
10063 00003CAB AB <1> stosd
10064 <1>
10065 <1> ; (total 40 bytes are written above as BIOS data state)
10066 <1>
10067 <1> bfn_svs_6:
10068 <1> ; 12/01/2021
10069 00003CAC F6C104 <1> test cl, 4
10070 00003CAF 7422 <1> jz short bfn_svs_8
10071 <1>
10072 <1> ; /* xxx: check this */
10073 <1> ; /* read/write mode dac */
10074 <1> ; write_byte(ES, BX, inb(VGAREG_DAC_STATE)); BX++;
10075 <1> ; /* pix address */
10076 <1> ; write_byte(ES, BX, inb(VGAREG_DAC_WRITE_ADDRESS)); BX++;
10077 <1> ; write_byte(ES, BX, inb(VGAREG_PEL_MASK)); BX++;
10078 <1> ; // Set the whole dac always, from 0
10079 <1> ; outb(VGAREG_DAC_WRITE_ADDRESS, 0x00);
10080 <1> ; for(i=0; i<256*3; i++) {
10081 <1> ; write_byte(ES, BX, inb(VGAREG_DAC_DATA)); BX++;
10082 <1> ; }
10083 <1> ; write_byte(ES, BX, 0); BX++; /* color select register */
10084 <1>
10085 <1> ; /* read/write mode dac */
10086 00003CB1 66BAC703 <1> mov dx, 3C7h ; VGAREG_DAC_STATE
10087 00003CB5 EC <1> in al, dx
10088 00003CB6 AA <1> stosb

```

```

10089      <1>      ; /* pix address */
10090      <1>      ;mov     dx, VGAREG_DAC_WRITE_ADDRESS ; 3C8h
10091      <1>      ;mov     dl, 0C8h
10092      <1>      inc     dl
10093      <1>      in      al, dx
10094      <1>      stosb
10095      <1>      ;mov     dx, VGAREG_PEL_MASK ; 3C6h
10096      <1>      mov     dl, 0C6h
10097      <1>      in      al, dx
10098      <1>      stosb
10099      <1>      ;// Set the whole dac always, from 0
10100      <1>      xor     al, al ; 0
10101      <1>      ;mov     dx, VGAREG_DAC_WRITE_ADDRESS ; 3C8h
10102      <1>      mov     dl, 0C8h
10103      <1>      out     dx, al
10104      <1>
10105      <1>      push    ecx ; 22/01/2021
10106      <1>      ;for(i=0;i<256*3;i++) {
10107      <1>      ;mov     ecx, 256*3 ; 768 bytes
10108      <1>      ; 03/08/2022
10109      <1>      sub     ecx, ecx
10110      <1>      mov     ch, 3
10111      <1>      ; ecx = 300h = 768
10112      <1>      ;mov     dx, VGAREG_DAC_DATA ; 3C9h
10113      <1>      ;mov     dl, 0C9h
10114      <1>      inc     dl ; dx = 3C9h
10115      <1> bfn_svs_7:
10116      <1>      in      al, dx
10117      <1>      stosb
10118      <1>      loop   bfn_svs_7
10119      <1>      pop     ecx ; 22/01/2021
10120      <1>
10121      <1>      ; /* color select register */
10122      <1>      sub     al, al ; 0
10123      <1>      stosb
10124      <1>
10125      <1>      ; (total 772 bytes are written above as DAC state)
10126      <1> bfn_svs_8:
10127      <1>      retn
10128      <1>
10129      <1> vbe_biosfn_save_video_state:
10130      <1>      ; 23/01/2021
10131      <1>      ; 13/01/2021
10132      <1>      ; 12/01/2021 (TRDOS 386 v2.0.3)
10133      <1>      ; (vbe.c)
10134      <1>
10135      <1>      ; modified registers: eax, edx, edi, ch
10136      <1>
10137      <1>      ; input: edi = state buffer address
10138      <1>      ; output:
10139      <1>      ;         VBE DISPI register contents will be saved
10140      <1>      ;         (18 bytes, 9 words)
10141      <1>
10142      <1>      ; outw(VBE_DISPI_IOPORT_INDEX,VBE_DISPI_INDEX_ENABLE);
10143      <1>      ; enable = inw(VBE_DISPI_IOPORT_DATA);
10144      <1>      ; write_word(ES, BX, enable);
10145      <1>      ; BX += 2;
10146      <1>      ; if (!(enable & VBE_DISPI_ENABLED))
10147      <1>      ;         return;
10148      <1>      ; for(i = VBE_DISPI_INDEX_XRES;
10149      <1>      ;         i <= VBE_DISPI_INDEX_Y_OFFSET; i++) {
10150      <1>      ;     if (i != VBE_DISPI_INDEX_ENABLE) {
10151      <1>      ;         outw(VBE_DISPI_IOPORT_INDEX, i);
10152      <1>      ;         write_word(ES, BX, inw(VBE_DISPI_IOPORT_DATA));
10153      <1>      ;         BX += 2;
10154      <1>      ;     }
10155      <1>      ; }
10156      <1>
10157      <1>      mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10158      <1>      ;mov     eax, 04h ; VBE_DISPI_INDEX_ENABLE
10159      <1>      ;03/08/2022
10160      <1>      out     dx, ax
10161      <1>      ;mov     dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10162      <1>      inc     dl
10163      <1>      in      ax, dx ; enable (status)
10164      <1>      stosw
10165      <1>      and     ax, 1 ; VBE_DISPI_ENABLED
10166      <1>      jnz     short vbe_bfn_svs_0
10167      <1>      ; 23/01/2021
10168      <1>      ; ax = 0
10169      <1>      ; VBE_DISPI_DISABLED
10170      <1>      ; 13/01/2021
10171      <1>      ; clear remain 8 bytes
10172      <1>      ;xor     eax, eax
10173      <1>      stosd ; 2
10174      <1>      stosd ; 2
10175      <1>      stosd ; 2
10176      <1>      stosd ; 2
10177      <1>      retn
10178      <1> vbe_bfn_svs_0:
10179      <1>      ; VBE_DISPI_ENABLED
10180      <1>
10181      <1>      ;sub     eax, eax
10182      <1>      sub     al, al ; eax = 0
10183      <1>
10184      <1>      ; from VBE_DISPI_INDEX_XRES
10185      <1>      ; to VBE_DISPI_INDEX_BPP
10186      <1>
10187      <1>      mov     ch, 3
10188      <1>      ; al = 0 ; VBE_DISPI_INDEX_XRES - 1
10189      <1>
10190      <1>      call    vbe_bfn_svs_1
10191      <1>
10192      <1>      ; from VBE_DISPI_INDEX_BANK
10193      <1>      ; to VBE_DISPI_INDEX_Y_OFFSET
10194      <1>
10195      <1>      inc     al
10196      <1>      ; al = 4 ; VBE_DISPI_INDEX_BANK - 1
10197      <1>
10198      <1>      mov     ch, 5
10199      <1> vbe_bfn_svs_1:
10200      <1>      inc     al ; from VBE_DISPI_INDEX_XRES
10201      <1>      ; to VBE_DISPI_INDEX_BPP
10202      <1>      ;mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10203      <1>      dec     dl ; 1CEh
10204      <1>      out     dx, ax
10205      <1>      push    eax
10206      <1>      ;mov     dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10207      <1>      inc     dl ; 1CFh
10208      <1>      in      ax, dx
10209      <1>      stosw
10210      <1>      pop     eax
10211      <1>      dec     ch
10212      <1>      jnz     short vbe_bfn_svs_1

```

```

10213 00003D0A C3      <1>      retn
10214                  <1>
10215                  <1> biosfn_restore_video_state:
10216                  <1>      ; 22/01/2021
10217                  <1>      ; 13/01/2021
10218                  <1>      ; 12/01/2021 (TRDOS 386 v2.0.3)
10219                  <1>      ; (vgabios.c)
10220                  <1>
10221                  <1>      ; modified registers: eax, edx, esi, edi, ch
10222                  <1>
10223                  <1>      ;mov     esi, VBE3SAVERESTOREBLOCK
10224                  <1>
10225                  <1>      ; input: esi = state buffer address
10226                  <1>
10227 00003D0B F6C101    <1>      test     cl, 1
10228 00003D0E 0F84A9000000 <1>      jz       bfn_rvs_6
10229                  <1>
10230 00003D14 66817E40D403 <1>      cmp     word [esi+64], 3D4h ; must be 3D4h
10231 00003D1A 7402      <1>      je       short bfn_rvs_0
10232                  <1>      ; it is seen as valid buffer
10233 00003D1C F9        <1>      stc
10234 00003D1D C3        <1>      retn
10235                  <1>
10236                  <1> bfn_rvs_0:
10237 00003D1E 89F7      <1>      mov     edi, esi ; addr1
10238 00003D20 83C605    <1>      add     esi, 5 ; skip 1st 5 bytes for now
10239                  <1>
10240                  <1>      ; // Reset Attribute Ctl flip-flop
10241                  <1>      ; inb(VGAREG_ACTL_RESET);
10242 00003D23 66BADA03 <1>      mov     dx, 3DAh ; VGAREG_ACTL_RESET
10243 00003D27 EC        <1>      in      al, dx
10244                  <1>
10245                  <1>      ; for(i=1;i<=4;i++){
10246 00003D28 B001      <1>      mov     al, 1
10247                  <1>      ; ;mov dx, VGAREG_SEQU_ADDRESS ; 3C4h
10248                  <1>      ; ;mov dl, 0C4h
10249 00003D2A B504      <1>      mov     ch, 4
10250                  <1> bfn_rvs_1:
10251                  <1>      ; outb(VGAREG_SEQU_ADDRESS, i);
10252                  <1>      ; mov dx, VGAREG_SEQU_ADDRESS ; 3C4h
10253 00003D2C B2C4      <1>      mov     dl, 0C4h
10254 00003D2E EE        <1>      out     dx, al
10255                  <1>      ; mov dx, VGAREG_SEQU_DATA ; 3C5h
10256 00003D2F FEC2      <1>      inc     dl ; dx = 3C5h
10257                  <1>      ; outb(VGAREG_SEQU_DATA)
10258                  <1>      push    eax
10259 00003D32 AC        <1>      lodsb   ; (4 bytes in loop)
10260 00003D33 EE        <1>      out     dx, al
10261 00003D34 58        <1>      pop     eax
10262                  <1>      ; mov dx, VGAREG_SEQU_ADDRESS ; 3C4h
10263                  <1>      ; dec dl
10264 00003D35 FEC0      <1>      inc     al ; i++
10265 00003D37 FECD      <1>      dec     ch
10266 00003D39 75F1      <1>      jnz     short bfn_rvs_1
10267                  <1>
10268                  <1>      ; outb(VGAREG_SEQU_ADDRESS, 0);
10269 00003D3B 28C0      <1>      sub     al, al ; 0
10270 00003D3D EE        <1>      out     dx, al
10271                  <1>      ; outb(VGAREG_SEQU_DATA)
10272                  <1>      ; mov dx, VGAREG_SEQU_DATA ; 3C5h
10273 00003D3E FEC2      <1>      inc     dl ; dx = 3C5h
10274 00003D40 AC        <1>      lodsb   ; (+1 byte)
10275 00003D41 EE        <1>      out     dx, al
10276                  <1>
10277                  <1>      ; // Disable CRTC write protection
10278                  <1>      ; outw(crtc_addr,0x0011);
10279                  <1>      ; mov dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
10280 00003D42 B2D4      <1>      mov     dl, 0D4h
10281 00003D44 66B81100 <1>      mov     ax, 11h
10282 00003D48 66EF      <1>      out     dx, ax
10283                  <1>
10284                  <1>      ; // Set CRTC regs
10285                  <1>
10286                  <1>      ; for(i=0;i<=0x18;i++) {
10287                  <1>      ;     if (i != 0x11) {
10288 00003D4A 28C0      <1>      sub     al, al ; 0
10289                  <1>      ; ;mov dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
10290                  <1>      ; ;mov dl, 0D4h
10291 00003D4C B519      <1>      mov     ch, 25
10292                  <1> bfn_rvs_2:
10293                  <1>      ; outb(crtc_addr,i);
10294                  <1>      ; mov dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
10295 00003D4E B2D4      <1>      mov     dl, 0D4h
10296 00003D50 EE        <1>      out     dx, al
10297                  <1>      ; mov dx, VGAREG_VGA_CRTC_DATA ; 3D5h
10298 00003D51 FEC2      <1>      inc     dl ; dx = 3D5h
10299                  <1>      ; inb(crtc_addr+1)
10300 00003D53 50        <1>      push    eax
10301 00003D54 AC        <1>      lodsb   ; (25 bytes in loop)
10302 00003D55 EE        <1>      out     dx, al
10303 00003D56 58        <1>      pop     eax
10304                  <1>      ; mov dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
10305                  <1>      ; dec dl
10306 00003D57 FEC0      <1>      inc     al ; i++
10307 00003D59 3C11      <1>      cmp     al, 17 ; 11h
10308 00003D5B 7505      <1>      jne     short bfn_rvs_3
10309 00003D5D AC        <1>      lodsb
10310 00003D5E 88C4      <1>      mov     ah, al ; *
10311 00003D60 B012      <1>      mov     al, 18
10312                  <1> bfn_rvs_3:
10313 00003D62 FECD      <1>      dec     ch
10314 00003D64 75E8      <1>      jnz     short bfn_rvs_2
10315                  <1>
10316                  <1>      ; // select crtc base address
10317                  <1>      ; v = inb(VGAREG_READ_MISC_OUTPUT) & ~0x01;
10318                  <1>      ; if (crtc_addr = 0x3d4)
10319                  <1>      ;     v |= 0x01;
10320                  <1>      ; outb(VGAREG_WRITE_MISC_OUTPUT, v);
10321                  <1>
10322                  <1>      ; ;mov dx, VGAREG_READ_MISC_OUTPUT ; 3CCh
10323                  <1>      ; mov dl, 0CCh
10324                  <1>      ; in al, dl
10325                  <1>      ; and al, 1
10326                  <1>      ; ;mov dx, VGAREG_WRITE_MISC_OUTPUT ; 3C2h
10327                  <1>      ; mov dl, 0C2h
10328                  <1>      ; or al, 1
10329                  <1>      ; out dx, al
10330                  <1>
10331                  <1>      ; // enable write protection if needed
10332                  <1>      ; outb(crtc_addr, 0x11);
10333                  <1>      ; outb(crtc_addr+1, read_byte(ES, BX - 0x18 + 0x11));
10334                  <1>      ; mov dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
10335 00003D66 B2D4      <1>      mov     dl, 0D4h
10336 00003D68 B011      <1>      mov     al, 11h

```

```

10337 00003D6A EE      <1>      out      dx, al
10338 00003D6B 88E0    <1>      mov      al, ah ; *
10339 00003D6D FEC2    <1>      inc      dl ; dx = 3D5h
10340 00003D6F EE      <1>      out      dx, al
10341                                <1>
10342                                <1>      ; // Set Attribute Ctl
10343 00003D70 8A6703    <1>      mov      ah, [edi+3] ; addr1+3, ah = ar_index
10344 00003D73 80E420    <1>      and      ah, 20h ; (ar_index & 0x20)
10345                                <1>
10346                                <1>      ; inb(VGAREG_ACTL_RESET);
10347                                <1>      ;mov      dx, 3DAh ; VGAREG_ACTL_RESET
10348 00003D76 B2DA      <1>      mov      dl, 0DAh
10349 00003D78 EC        <1>      in       al, dx
10350                                <1>
10351                                <1>      ; for(i=0;i<=0x13;i++) {
10352 00003D79 28C0      <1>      sub      al, al ; 0
10353 00003D7B B514      <1>      mov      ch, 20
10354                                <1>      bfn_rvs_4:
10355                                <1>      ; outb(VGAREG_ACTL_ADDRESS, i | (ar_index & 0x20));
10356 00003D7D 50        <1>      push     eax
10357 00003D7E 08E0      <1>      or       al, ah
10358                                <1>      ;mov      dx, VGAREG_ACTL_ADDRESS ; 3C0h
10359 00003D80 B2C0      <1>      mov      dl, 0C0h
10360 00003D82 EE        <1>      out      dx, al
10361                                <1>      ;mov      dx, VGAREG_ACTL_WRITE_DATA ; 3C0h
10362                                <1>      ;mov      dl, 0C0h
10363 00003D83 AC        <1>      lodsb     ; (20 bytes in loop)
10364 00003D84 EE        <1>      out      dx, al
10365 00003D85 58        <1>      pop      eax
10366 00003D86 FEC0      <1>      inc      al ; i++
10367 00003D88 FECD      <1>      dec      ch
10368 00003D8A 75F1      <1>      jnz      short bfn_rvs_4
10369                                <1>
10370                                <1>      ; outb(VGAREG_ACTL_ADDRESS, ar_index);
10371                                <1>      ;mov      dx, VGAREG_ACTL_ADDRESS ; 3C0h
10372                                <1>      ;mov      dl, 0C0h
10373 00003D8C 88E0      <1>      mov      al, ah ; ar_index
10374 00003D8E EE        <1>      out      dx, al
10375                                <1>
10376                                <1>      ; inb(VGAREG_ACTL_RESET);
10377                                <1>      ;mov      dx, VGAREG_ACTL_RESET ; 3DAh
10378 00003D8F B2DA      <1>      mov      dl, 0DAh
10379 00003D91 EC        <1>      in       al, dx
10380                                <1>
10381                                <1>      ; for(i=0;i<=8;i++) {
10382 00003D92 28C0      <1>      sub      al, al ; 0
10383                                <1>      ;mov      dx, VGAREG_GRDC_ADDRESS ; 3CEh
10384                                <1>      ;mov      dl, 0CEh
10385 00003D94 B509      <1>      mov      ch, 9
10386                                <1>      bfn_rvs_5:
10387                                <1>      ; outb(VGAREG_GRDC_ADDRESS,i)
10388                                <1>      ;mov      dx, VGAREG_GRDC_ADDRESS ; 3CEh
10389 00003D96 B2CE      <1>      mov      dl, 0CEh
10390 00003D98 EE        <1>      out      dx, al
10391                                <1>      ; outb(VGAREG_ACTL_READ_DATA)
10392 00003D99 50        <1>      push     eax
10393                                <1>      ;mov      dx, VGAREG_GRDC_DATA ; 3CFh
10394                                <1>      ;mov      dl, 0CFh
10395 00003D9A FEC2      <1>      inc      dl
10396 00003D9C AC        <1>      lodsb     ; (9 bytes in loop)
10397 00003D9D EE        <1>      out      dx, al
10398 00003D9E 58        <1>      pop      eax
10399                                <1>      ;dec      dl
10400 00003D9F FEC0      <1>      inc      al ; i++
10401 00003DA1 FECD      <1>      dec      ch
10402 00003DA3 75F1      <1>      jnz      short bfn_rvs_5
10403                                <1>
10404                                <1>      ; BX += 2; /* crtc_addr */ ; 3D4h
10405                                <1>      ; BX += 4; /* plane latches */ ; 0
10406 00003DA5 83C606    <1>      add      esi, 6
10407 00003DA8 56        <1>      push     esi ; *
10408                                <1>
10409                                <1>      ;outb(VGAREG_SEQU_ADDRESS, read_byte(ES, addr1)); addr1++;
10410                                <1>      ;outb(crtc_addr, read_byte(ES, addr1)); addr1++;
10411                                <1>      ;outb(VGAREG_GRDC_ADDRESS, read_byte(ES, addr1)); addr1++;
10412                                <1>      ;addr1++;
10413                                <1>      ;outb(crtc_addr - 0x4 + 0xa, read_byte(ES, addr1)); addr1++;
10414                                <1>
10415 00003DA9 89FE      <1>      mov      esi, edi ; start of state buffer
10416                                <1>
10417                                <1>      ;mov      dx, VGAREG_SEQU_ADDRESS ; 3C7h
10418 00003DAB B2C7      <1>      mov      dl, 0C7h
10419 00003DAD AC        <1>      lodsb
10420 00003DAE EE        <1>      out      dx, al
10421                                <1>      ;mov      dx, VGAREG_VGA_CRTC_ADDRESS ; 3D4h
10422 00003DAF B2D4      <1>      mov      dl, 0D4h
10423 00003DB1 AC        <1>      lodsb
10424 00003DB2 EE        <1>      out      dx, al
10425                                <1>      ;mov      dx, VGAREG_GRDC_ADDRESS ; 3CEh
10426 00003DB3 B2CE      <1>      mov      dl, 0CEh
10427 00003DB5 AC        <1>      lodsb
10428 00003DB6 EE        <1>      out      dx, al
10429 00003DB7 AC        <1>      lodsb ; addr1++
10430                                <1>      ;mov      dx, VGAREG_VGA_WRITE_FEATURE_CTL ; 3DAh
10431 00003DB8 B2DA      <1>      mov      dl, 0DAh
10432 00003DBA AC        <1>      lodsb
10433 00003DBB EE        <1>      out      dx, al
10434                                <1>
10435 00003DBC 5E        <1>      pop      esi ; *
10436                                <1>
10437                                <1>      ; (total 70 bytes are read above as controller hardware state)
10438                                <1>
10439                                <1>      bfn_rvs_6:
10440                                <1>      ; 13/01/2021
10441 00003DBD F6C102    <1>      test     cl, 2
10442 00003DC0 747D      <1>      jz       short bfn_rvs_9
10443                                <1>
10444                                <1>      ; VIDEO BIOS DATA
10445                                <1>      ; !!! this data is valid for TRDOS 386 v2 kernel only !!!
10446                                <1>      ; (this is not same with BOCHS/PLEX86 video bios, BIOS data)
10447                                <1>
10448                                <1>      ; if (CX & 2) {
10449                                <1>      ;write_byte(BIOSMEM_SEG,BIOSMEM_CURRENT_MODE, read_byte(ES, BX)); BX++;
10450                                <1>      ;write_word(BIOSMEM_SEG,BIOSMEM_NB_COLS, read_word(ES, BX)); BX += 2;
10451                                <1>      ;write_word(BIOSMEM_SEG,BIOSMEM_PAGE_SIZE, read_word(ES, BX)); BX += 2;
10452                                <1>      ;write_word(BIOSMEM_SEG,BIOSMEM_CRTC_ADDRESS, read_word(ES, BX)); BX += 2;
10453                                <1>      ;write_byte(BIOSMEM_SEG,BIOSMEM_NB_ROWS, read_byte(ES, BX)); BX++;
10454                                <1>      ;write_word(BIOSMEM_SEG,BIOSMEM_CHAR_HEIGHT, read_word(ES, BX)); BX += 2;
10455                                <1>      ;write_byte(BIOSMEM_SEG,BIOSMEM_VIDEO_CTL, read_byte(ES, BX)); BX++;
10456                                <1>      ;write_byte(BIOSMEM_SEG,BIOSMEM_SWITCHES, read_byte(ES, BX)); BX++;
10457                                <1>      ;write_byte(BIOSMEM_SEG,BIOSMEM_MODESET_CTL, read_byte(ES, BX)); BX++;
10458                                <1>      ;write_word(BIOSMEM_SEG,BIOSMEM_CURSOR_TYPE, read_word(ES, BX)); BX += 2;
10459                                <1>      ;for(i=0;i<8;i++) {
10460                                <1>      ;      write_word(BIOSMEM_SEG, BIOSMEM_CURSOR_POS+2*i, read_word(ES, BX));

```

```

10461      <1>      ; BX += 2;
10462      <1>      ;}
10463      <1>      ;write_word(BIOSMEM_SEG,BIOSMEM_CURRENT_START, read_word(ES, BX)); BX += 2;
10464      <1>      ;write_byte(BIOSMEM_SEG,BIOSMEM_CURRENT_PAGE, read_byte(ES, BX)); BX++;
10465      <1>      ;/* current font */
10466      <1>      ;write_word(0, 0x1f * 4, read_word(ES, BX)); BX += 2;
10467      <1>      ;write_word(0, 0x1f * 4 + 2, read_word(ES, BX)); BX += 2;
10468      <1>      ;write_word(0, 0x43 * 4, read_word(ES, BX)); BX += 2;
10469      <1>      ;write_word(0, 0x43 * 4 + 2, read_word(ES, BX)); BX += 2;
10470      <1>
10471      <1>      ; !!! save TRDOS 386 v2 kernel specific video bios data !!!
10472      <1>      ; (which is/are used by 'SET_MODE' function and/or it's sub functions)
10473      <1>
10474      00003DC2 66AD      <1>      lodsw      ; CRTC_ADDR, always 3D4h (color VGA) for TRDOS 386 v2
10475      <1>      ; skip 3D4h check if it is already checked
10476      00003DC4 F6C101    <1>      test       cl, 1
10477      00003DC7 7508      <1>      jnz        short bfn_rvs_7
10478      00003DC9 663DD403  <1>      cmp        ax, 3D4h
10479      00003DCD 7402      <1>      je         short bfn_rvs_7
10480      00003DCF F9       <1>      stc
10481      00003DD0 C3       <1>      retn
10482      <1> bfn_rvs_7:
10483      00003DD1 AC       <1>      lodsb
10484      00003DD2 A2[1E680000] <1>      mov        [CRT_MODE], al ; Current video mode (0FFh for VESA VBE modes)
10485      00003DD7 AC       <1>      lodsb
10486      00003DD8 A2[1F680000] <1>      mov        [CRT_MODE_SET], al ; 29h for mode 03h ; TRDOS 386 feature only !
10487      00003DDD 66AD      <1>      lodsw
10488      00003DDF 66A3[0E9D0100] <1>      mov        [video_mode], ax ; Current VESA VBE (SVGA, extended VGA) mode
10489      00003DE5 66AD      <1>      lodsw      ; (valid if [CRT_MODE] = 0FFh)
10490      00003DE7 66A3[38830100] <1>      mov        [CRT_LEN], ax ; page size (in bytes)
10491      00003DED 66AD      <1>      lodsw
10492      00003DEF 66A3[BC760100] <1>      mov        [CRT_START], ax ; video page start offset
10493      00003DF5 AC       <1>      lodsb
10494      00003DF6 A2[20680000] <1>      mov        [CRT_COLS], al ; nbcols, characters per row
10495      00003DFB AC       <1>      lodsb
10496      00003DFC A2[26680000] <1>      mov        [VGA_ROWS], al ; nbrows, (character) rows per page (not rows-1)
10497      00003E01 AC       <1>      lodsb
10498      00003E02 A2[22680000] <1>      mov        [CHAR_HEIGHT], al ; character font height (8 or 16 or 14)
10499      00003E07 AC       <1>      lodsb
10500      00003E08 A2[23680000] <1>      mov        [VGA_VIDEO_CTL], al ; ROM BIOS DATA AREA Offset 87h
10501      00003E0D AC       <1>      lodsb
10502      00003E0E A2[24680000] <1>      mov        [VGA_SWITCHES], al ; feature bit switches
10503      00003E13 AC       <1>      lodsb
10504      00003E14 A2[25680000] <1>      mov        [VGA_MODESET_CTL], al ; basic mode set options
10505      <1>      ; followings are only used by TRDOS 386 v2 (IBM PC/AT ROMBIOS) code
10506      <1>      ; (bochs/plex86 does not use and return those)
10507      00003E19 AC       <1>      lodsb
10508      00003E1A A2[21680000] <1>      mov        [CRT_PALETTE], al ; current color palette ; TRDOS 386 feature only !
10509      00003E1F AC       <1>      lodsb
10510      00003E20 A2[CE760100] <1>      mov        [ACTIVE_PAGE], al ; current video page
10511      00003E25 66AD      <1>      lodsw
10512      00003E27 66A3[37680000] <1>      mov        [CURSOR_MODE], ax ; cursor type
10513      <1>      ;lodsd
10514      <1>      ;mov      [CURSOR_POSN], eax ; cursor position for video page 0 and 1
10515      <1>      ;lodsd
10516      <1>      ;mov      [CURSOR_POSN+4], eax ; cursor position for video page 2 and 3
10517      <1>      ;lodsd
10518      <1>      ;mov      [CURSOR_POSN+8], eax ; cursor position for video page 4 and 5
10519      <1>      ;lodsd
10520      <1>      ;mov      [CURSOR_POSN+12], eax ; cursor position for video page 6 and 7
10521      00003E2D B504      <1>      mov        ch, 4
10522      00003E2F BF[BE760100] <1>      mov        edi, CURSOR_POSN
10523      <1> bfn_rvs_8:
10524      00003E34 A5       <1>      movsd
10525      00003E35 FECD      <1>      dec        ch
10526      00003E37 75FB      <1>      jnz        short bfn_rvs_8
10527      <1>      ; (font addr) protected mode address in kernel's/system memory space
10528      <1>      ; (not accessible/meaningful address value by user)
10529      00003E39 AD       <1>      lodsd
10530      00003E3A A3[4A830100] <1>      mov        [VGA_INT43H], eax ; VGA current (default) font address
10531      <1>
10532      <1>      ; (total 40 bytes are read&written above as BIOS data state)
10533      <1> bfn_rvs_9:
10534      <1>      ; 13/01/2021
10535      00003E3F F6C104    <1>      test       cl, 4
10536      00003E42 7421      <1>      jz         short bfn_rvs_11
10537      <1>
10538      <1>      ;BX++;
10539      <1>      ;v = read_byte(ES, BX); BX++;
10540      <1>      ;outb(VGAREG_PEL_MASK, read_byte(ES, BX)); BX++;
10541      <1>      ;// Set the whole dac always, from 0
10542      <1>      ;outb(VGAREG_DAC_WRITE_ADDRESS,0x00);
10543      <1>      ;for(i=0;i<256*3;i++) {
10544      <1>      ;    outb(VGAREG_DAC_DATA, read_byte(ES, BX)); BX++;
10545      <1>      ;}
10546      <1>      ;BX++;
10547      <1>      ;outb(VGAREG_DAC_WRITE_ADDRESS, v);
10548      <1>
10549      <1>      ; /* read/write mode dac */
10550      00003E44 AC       <1>      lodsb      ; skip ; VGAREG_DAC_STATE
10551      00003E45 AC       <1>      lodsb
10552      00003E46 88C4      <1>      mov        ah, al ; * ; v
10553      00003E48 AC       <1>      lodsb
10554      00003E49 66BAC603  <1>      mov        dx, VGAREG_PEL_MASK ; 3C6h
10555      00003E4D EE       <1>      out        dx, al
10556      <1>      ;// Set the whole dac always, from 0
10557      00003E4E 30C0      <1>      xor        al, al ; 0
10558      <1>      ;mov        dx, VGAREG_DAC_WRITE_ADDRESS ; 3C8h
10559      00003E50 B2C8      <1>      mov        dl, 0C8h
10560      00003E52 EE       <1>      out        dx, al
10561      <1>
10562      00003E53 51       <1>      push       ecx ; 22/01/2021
10563      <1>      ;for(i=0;i<256*3;i++) {
10564      <1>      ;mov        ecx, 256*3 ; 768 bytes
10565      <1>      ; 03/08/2022
10566      00003E54 29C9      <1>      sub        ecx, ecx
10567      00003E56 B503      <1>      mov        ch, 3
10568      <1>      ; ecx = 300h = 768
10569      <1>      ;mov        dx, VGAREG_DAC_DATA ; 3C9h
10570      <1>      ;mov        dl, 0C9h
10571      00003E58 FEC2      <1>      inc        dl ; dx = 3C9h
10572      <1> bfn_rvs_10:
10573      00003E5A AC       <1>      lodsb
10574      00003E5B EE       <1>      out        dx, al
10575      00003E5C E2FC      <1>      loop       bfn_rvs_10
10576      00003E5E 59       <1>      pop        ecx ; 22/01/2021
10577      <1>
10578      <1>      ; /* color select register */
10579      00003E5F AC       <1>      lodsb      ; skip
10580      <1>
10581      00003E60 88E0      <1>      mov        al, ah ; * ; v
10582      <1>
10583      <1>      ;mov        dx, VGAREG_DAC_WRITE_ADDRESS ; 3C8h
10584      <1>      ;mov        dl, 0C8h

```

```

10585 00003E62 FECA      <1>      dec    dl ; dx = 3C8h
10586 00003E64 EE        <1>      out    dx, al ; * ; v
10587                    <1>
10588                    <1>      ; (total 772 bytes are read above as DAC state)
10589                    <1> bfn_rvs_11:
10590 00003E65 C3         <1>      retn
10591                    <1>
10592                    <1> vbe_biosfn_restore_video_state:
10593                    <1>      ; 23/01/2021
10594                    <1>      ; 13/01/2021 (TRDOS 386 v2.0.3)
10595                    <1>      ; (vbe.c)
10596                    <1>
10597                    <1>      ; modified registers: eax, edx, esi, ch
10598                    <1>
10599                    <1>      ; input: esi = state buffer address
10600                    <1>      ; output:
10601                    <1>      ;       VBE DISPI register contents will be restored
10602                    <1>      ;       (18 bytes, 9 words)
10603                    <1>
10604                    <1>      ; enable = read_word(ES, BX);
10605                    <1>      ; BX += 2;
10606                    <1>      ;
10607                    <1>      ; if (!(enable & VBE_DISPI_ENABLED)) {
10608                    <1>      ;       outw(VBE_DISPI_IOPORT_INDEX, VBE_DISPI_INDEX_ENABLE);
10609                    <1>      ;       outw(VBE_DISPI_IOPORT_DATA, enable);
10610                    <1>      ; } else {
10611                    <1>      ;       outw(VBE_DISPI_IOPORT_INDEX, VBE_DISPI_INDEX_XRES);
10612                    <1>      ;       outw(VBE_DISPI_IOPORT_DATA, read_word(ES, BX));
10613                    <1>      ;       BX += 2;
10614                    <1>      ;       outw(VBE_DISPI_IOPORT_INDEX, VBE_DISPI_INDEX_YRES);
10615                    <1>      ;       outw(VBE_DISPI_IOPORT_DATA, read_word(ES, BX));
10616                    <1>      ;       BX += 2;
10617                    <1>      ;       outw(VBE_DISPI_IOPORT_INDEX, VBE_DISPI_INDEX_BPP);
10618                    <1>      ;       outw(VBE_DISPI_IOPORT_DATA, read_word(ES, BX));
10619                    <1>      ;       BX += 2;
10620                    <1>      ;       outw(VBE_DISPI_IOPORT_INDEX, VBE_DISPI_INDEX_ENABLE);
10621                    <1>      ;       outw(VBE_DISPI_IOPORT_DATA, enable);
10622                    <1>      ;
10623                    <1>      ;       for(i = VBE_DISPI_INDEX_BANK; i <= VBE_DISPI_INDEX_Y_OFFSET; i++)
10624                    <1>      ;       {
10625                    <1>      ;           outw(VBE_DISPI_IOPORT_INDEX, i);
10626                    <1>      ;           outw(VBE_DISPI_IOPORT_DATA, read_word(ES, BX));
10627                    <1>      ;           BX += 2;
10628                    <1>      ;       }
10629                    <1>      ; }
10630                    <1>
10631 00003E66 66AD         <1>      lodsw   ; enable (status, enabled=1, disabled=0)
10632 00003E68 66BACE01    <1>      mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10633                    <1>      ; 23/01/2021
10634 00003E6C 6683E001    <1>      and     ax, 1 ; VBE_DISPI_ENABLED
10635 00003E70 750B         <1>      jnz     short vbe_bfn_rvs_1
10636                    <1>      ; ax = 0
10637                    <1>      ; VBE_DISPI_DISABLED
10638                    <1> vbe_bfn_rvs_0:
10639                    <1>      ; enable (disable) dispi
10640                    <1>      ; dx = 01CEh ; VBE_DISPI_IOPORT_INDEX
10641                    <1>      ; ah = 0
10642 00003E72 50          <1>      push    eax
10643 00003E73 B004         <1>      mov     al, 04h ; VBE_DISPI_INDEX_ENABLE
10644 00003E75 66EF         <1>      out     dx, ax
10645                    <1>      ; mov     dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10646 00003E77 FEC2         <1>      inc     dl
10647 00003E79 58          <1>      pop     eax
10648 00003E7A 66EF         <1>      out     dx, ax ; enable (or disable)
10649 00003E7C C3          <1>      retn
10650                    <1> vbe_bfn_rvs_1:
10651                    <1>      ; VBE_DISPI_ENABLED
10652                    <1>
10653                    <1>      ; from VBE_DISPI_INDEX_XRES
10654                    <1>      ; to VBE_DISPI_INDEX_BPP
10655                    <1>
10656 00003E7D B503         <1>      mov     ch, 3
10657 00003E7F 28C0         <1>      sub     al, al ; 0 ; VBE_DISPI_INDEX_XRES - 1
10658                    <1>      ; ax = 0
10659                    <1>
10660 00003E81 E80B000000    <1>      call    vbe_bfn_rvs_2
10661                    <1>
10662                    <1>      ; outw(VBE_DISPI_IOPORT_INDEX, VBE_DISPI_INDEX_ENABLE);
10663                    <1>      ; outw(VBE_DISPI_IOPORT_DATA, enable);
10664                    <1>
10665                    <1>      ; 23/01/2021
10666 00003E86 B001         <1>      mov     al, 1 ; VBE_DISPI_ENABLED
10667                    <1>      ; ax = 1
10668 00003E88 E8E5FFFFFF    <1>      call    vbe_bfn_rvs_0
10669                    <1>
10670                    <1>      ; from VBE_DISPI_INDEX_BANK
10671                    <1>      ; to VBE_DISPI_INDEX_Y_OFFSET
10672                    <1>
10673 00003E8D B505         <1>      mov     ch, 5
10674                    <1>      ; 23/01/2021
10675 00003E8F B004         <1>      mov     al, 4 ; VBE_DISPI_INDEX_BANK - 1
10676                    <1>      ; ax = 4
10677                    <1> vbe_bfn_rvs_2:
10678                    <1>      inc     al ; from VBE_DISPI_INDEX_XRES
10679                    <1>      ; to VBE_DISPI_INDEX_BPP
10680                    <1>      ; mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10681                    <1>      ; mov     dl, 0CEh
10682 00003E93 66EF         <1>      out     dx, ax
10683 00003E95 50          <1>      push    eax
10684                    <1>      ; mov     dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10685 00003E96 FEC2         <1>      inc     dl ; 1CFh
10686 00003E98 66AD         <1>      lodsw
10687 00003E9A 66EF         <1>      out     dx, ax
10688 00003E9C 58          <1>      pop     eax
10689 00003E9D FECA         <1>      dec     dl ; 1CEh
10690 00003E9F FECD         <1>      dec     ch
10691 00003EA1 75EE         <1>      jnz     short vbe_bfn_rvs_2
10692 00003EA3 C3          <1>      retn
10693                    <1>
10694                    <1> ; -----
10695                    <1>
10696                    <1> dispi_set_enable:
10697                    <1>      ; 03/08/2022
10698                    <1>      ; 23/11/2020
10699                    <1>      ; Input:
10700                    <1>      ;       ax = VBE_DISPI_ENABLED = 1
10701                    <1>      ;       or VBE_DISPI_DISABLED = 0
10702                    <1>      ;
10703                    <1>      ; Modified registers: none
10704                    <1>
10705                    <1>      ; push    edx
10706                    <1>      ; push    eax
10707                    <1>      ; mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10708                    <1>      ; mov     ax, 04h ; VBE_DISPI_INDEX_ENABLE

```



```

10709      <1>      ;out    dx, ax
10710      <1>      ;pop    eax
10711      <1>      ;;mov   dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10712      <1>      ;;mov   dl, 0CFh
10713      <1>      ;inc    dl
10714      <1>      ;out    dx, ax
10715      <1>      ;pop    edx
10716      <1>      ;retn
10717      <1>
10718      <1>      ; 25/11/2020
10719      <1>      ; Modified registers: edx
10720      <1>      ;;push   edx
10721      <1>      ;mov    dx, 04h ; VBE_DISPI_INDEX_ENABLE
10722      <1>      ; 03/08/2022
10723      <1>      sub     dh, dh
10724      <1>      mov     dl, 04h ; VBE_DISPI_INDEX_ENABLE
10725      <1>
10726      <1>      ;;call   dispi_set_parms
10727      <1>      ;;pop    edx
10728      <1>      ;;retn
10729      <1>      ;jmp     short dispi_set_parms
10730      <1>
10731      <1>      dispi_set_parms:
10732      <1>      ; 03/08/2022
10733      <1>      ; 25/11/2020
10734      <1>      ; Input:
10735      <1>      ;     ax = data
10736      <1>      ;     dx = vbe dispi register index
10737      <1>      ;
10738      <1>      ; Modified registers: edx
10739      <1>
10740      <1>      push     eax
10741      <1>      ;mov     ax, dx
10742      <1>      ; 03/08/2022
10743      <1>      mov     eax, edx
10744      <1>      mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10745      <1>      out     dx, ax
10746      <1>      pop     eax
10747      <1>      ;mov     dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10748      <1>      ;mov     dl, 0CFh
10749      <1>      inc     dl
10750      <1>      out     dx, ax
10751      <1>      retn
10752      <1>
10753      <1>      dispi_set_bpp:
10754      <1>      ; 03/08/2022
10755      <1>      ; 25/11/2020
10756      <1>      ; Input:
10757      <1>      ;     ax = Bits per pixel value
10758      <1>      ;     (8,16,24,32)
10759      <1>      ;
10760      <1>      ; Modified registers: none
10761      <1>
10762      <1>      ;push    edx
10763      <1>      ;push    eax
10764      <1>      ;mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10765      <1>      ;mov     ax, 03h ; VBE_DISPI_INDEX_BPP
10766      <1>      ;out     dx, ax
10767      <1>      ;pop     eax
10768      <1>      ;;mov    dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10769      <1>      ;;mov    dl, 0CFh
10770      <1>      ;inc     dl
10771      <1>      ;out     dx, ax
10772      <1>      ;pop     edx
10773      <1>      ;retn
10774      <1>
10775      <1>      ; 25/11/2020
10776      <1>      ; Modified registers: edx
10777      <1>      ;;push   edx
10778      <1>      ;mov     dx, 03h ; VBE_DISPI_INDEX_BPP
10779      <1>      ; 03/08/2022
10780      <1>      sub     dh, dh
10781      <1>      mov     dl, 03h ; VBE_DISPI_INDEX_BPP
10782      <1>
10783      <1>      ;;call   dispi_set_parms
10784      <1>      ;;pop    edx
10785      <1>      ;;retn
10786      <1>      ;jmp     short dispi_set_parms
10787      <1>
10788      <1>      dispi_set_xres:
10789      <1>      ; 03/08/2022
10790      <1>      ; 25/11/2020
10791      <1>      ; Input:
10792      <1>      ;     ax = X resolution (screen width)
10793      <1>      ;     (320,640,800,1024,1280,1920)
10794      <1>      ;
10795      <1>      ; Modified registers: none
10796      <1>
10797      <1>      ;push    edx
10798      <1>      ;push    eax
10799      <1>      ;mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10800      <1>      ;mov     ax, 01h ; VBE_DISPI_INDEX_XRES
10801      <1>      ;out     dx, ax
10802      <1>      ;pop     eax
10803      <1>      ;;mov    dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10804      <1>      ;;mov    dl, 0CFh
10805      <1>      ;inc     dl
10806      <1>      ;out     dx, ax
10807      <1>      ;pop     edx
10808      <1>      ;retn
10809      <1>
10810      <1>      ; 25/11/2020
10811      <1>      ; Modified registers: edx
10812      <1>      ;;push   edx
10813      <1>      ;mov     dx, 01h ; VBE_DISPI_INDEX_XRES
10814      <1>      ; 03/08/2022
10815      <1>      sub     dh, dh
10816      <1>      mov     dl, 01h ; VBE_DISPI_INDEX_XRES
10817      <1>      ;;call   dispi_set_parms
10818      <1>      ;;pop    edx
10819      <1>      ;;retn
10820      <1>      ;jmp     short dispi_set_parms
10821      <1>
10822      <1>      dispi_set_yres:
10823      <1>      ; 03/08/2022
10824      <1>      ; 25/11/2020
10825      <1>      ; Input:
10826      <1>      ;     ax = Y resolution (screen height)
10827      <1>      ;     (200,400,600,720,768,1080)
10828      <1>      ;
10829      <1>      ; Modified registers: none
10830      <1>
10831      <1>      ;push    edx
10832      <1>      ;push    eax

```

```

10833      <1>      ;mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10834      <1>      ;mov     ax, 02h  ; VBE_DISPI_INDEX_YRES
10835      <1>      ;out      dx, ax
10836      <1>      ;pop      eax
10837      <1>      ;;mov     dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10838      <1>      ;;mov     dl, 0CFh
10839      <1>      ;inc      dl
10840      <1>      ;out      dx, ax
10841      <1>      ;pop      edx
10842      <1>      ;retn
10843      <1>
10844      <1>      ; 25/11/2020
10845      <1>      ; Modified registers: edx
10846      <1>      ;;push    edx
10847      <1>      ;mov     dx, 02h  ; VBE_DISPI_INDEX_YRES
10848      <1>      ; 03/08/2022
10849      <1>      sub      dh, dh
10850      <1>      mov     dl, 02h  ; VBE_DISPI_INDEX_YRES
10851      <1>      ;;call    dispi_set_parms
10852      <1>      ;;pop     edx
10853      <1>      ;;retn
10854      <1>      jmp     short dispi_set_parms
10855      <1>
10856      <1>      dispi_set_bank:
10857      <1>      ; 03/08/2022
10858      <1>      ; 25/11/2020
10859      <1>      ; Input:
10860      <1>      ;         ax = video memory bank number
10861      <1>      ;
10862      <1>      ; Modified registers: none
10863      <1>
10864      <1>      ;push    edx
10865      <1>      ;push    eax
10866      <1>      ;mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10867      <1>      ;mov     ax, 05h  ; VBE_DISPI_INDEX_BANK
10868      <1>      ;out      dx, ax
10869      <1>      ;pop      eax
10870      <1>      ;;mov     dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10871      <1>      ;;mov     dl, 0CFh
10872      <1>      ;inc      dl
10873      <1>      ;out      dx, ax
10874      <1>      ;pop      edx
10875      <1>      ;retn
10876      <1>
10877      <1>      ; 25/11/2020
10878      <1>      ; Modified registers: edx
10879      <1>      ;;push    edx
10880      <1>      ;mov     dx, 05h  ; VBE_DISPI_INDEX_BANK
10881      <1>      ; 03/08/2022
10882      <1>      sub      dh, dh
10883      <1>      mov     dl, 05h  ; VBE_DISPI_INDEX_BANK
10884      <1>      ;;call    dispi_set_parms
10885      <1>      ;;pop     edx
10886      <1>      ;;retn
10887      <1>      jmp     short dispi_set_parms
10888      <1>
10889      <1>      dispi_get_enable:
10890      <1>      ; 03/08/2022
10891      <1>      ; 27/11/2020
10892      <1>      ; Input:
10893      <1>      ;         none
10894      <1>      ; Output:
10895      <1>      ;         ax = vbe dispi status
10896      <1>      ;
10897      <1>      ; Modified registers: eax
10898      <1>
10899      <1>      ;push    edx
10900      <1>      ;mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10901      <1>      ;mov     ax, 04h  ; VBE_DISPI_INDEX_ENABLE
10902      <1>      ;out      dx, ax
10903      <1>      ;;mov     dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10904      <1>      ;;mov     dl, 0CFh
10905      <1>      ;inc      dl
10906      <1>      ;in       ax, dx
10907      <1>      ;pop      edx
10908      <1>      ;retn
10909      <1>
10910      <1>      ; 27/11/2020
10911      <1>      ; Modified registers: eax, edx
10912      <1>      ;;push    edx
10913      <1>      ;mov     ax, 04h  ; VBE_DISPI_INDEX_ENABLE
10914      <1>      ; 03/08/2022
10915      <1>      sub      ah, ah
10916      <1>      mov     al, 04h  ; VBE_DISPI_INDEX_ENABLE
10917      <1>
10918      <1>      ;;call    dispi_get_parms
10919      <1>      ;;pop     edx
10920      <1>      ;;retn
10921      <1>      ;jmp     short dispi_get_parms
10922      <1>
10923      <1>      dispi_get_parms:
10924      <1>      ; 25/11/2020
10925      <1>      ; Input:
10926      <1>      ;         ax = vbe dispi register index
10927      <1>      ; output:
10928      <1>      ;         ax = data
10929      <1>      ;
10930      <1>      ; Modified registers: eax, edx
10931      <1>
10932      <1>      mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
10933      <1>      out      dx, ax
10934      <1>      ;mov     dx, 01CFh ; VBE_DISPI_IOPORT_DATA
10935      <1>      ;mov     dl, 0CFh
10936      <1>      inc      dl
10937      <1>      in       ax, dx
10938      <1>      ;retn
10939      <1>
10940      <1>      vga_compat_setup:
10941      <1>      ; 03/08/2022
10942      <1>      ; 26/11/2020
10943      <1>      ; 25/11/2020
10944      <1>      ; VGA compatibility setup
10945      <1>      ; (vbe.c, 02/01/2020, vruppert)
10946      <1>      ;
10947      <1>      ; Input:
10948      <1>      ;         none
10949      <1>      ;
10950      <1>      ; Modified registers: eax, edx
10951      <1>
10952      <1>      ; 26/11/2020
10953      <1>      ;push    eax
10954      <1>      ;push    edx
10955      <1>
10956      <1>      ; set CRT x resolution

```

```

10957 00003EDE 66BACE01 <1> mov dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
10958 <1> ;mov ax, 01h ; VBE_DISPI_INDEX_XRES
10959 <1> ; 03/08/2022
10960 00003EE2 31C0 <1> xor eax, eax
10961 00003EE4 FEC0 <1> inc al
10962 <1> ; eax = 1
10963 00003EE6 66EF <1> out dx, ax
10964 <1> ;mov dx, 1CFh ; VBE_DISPI_IOPORT_DATA
10965 00003EE8 FEC2 <1> inc dl
10966 00003EEA 66ED <1> in ax, dx
10967 00003EEC 50 <1> push eax
10968 00003EED 66BAD403 <1> mov dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
10969 00003EF1 66B81100 <1> mov ax, 0011h ; Vertical retrace end register
10970 00003EF5 66EF <1> out dx, ax
10971 <1> ;pop eax
10972 <1> ;push eax
10973 00003EF7 8B0424 <1> mov eax, [esp]
10974 <1> ;shr ax, 3 ; / 8 for pixel to character
10975 <1> ;dec ax ; - 1 (EGA or VGA?)
10976 <1> ; 03/08/2022
10977 00003EFA C1E803 <1> shr eax, 3
10978 00003EFD 48 <1> dec eax
10979 00003EFE 88C4 <1> mov ah, al
10980 00003F00 B001 <1> mov al, 01h ; Horizontal display end register
10981 00003F02 66EF <1> out dx, ax
10982 00003F04 58 <1> pop eax
10983 <1>
10984 00003F05 E89C000000 <1> call vga_set_virt_width
10985 <1>
10986 <1> ; set CRT Y resolution
10987 <1> ; 03/08/2022
10988 00003F0A 66BACE01 <1> mov dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
10989 00003F0E 66B80200 <1> mov ax, 02h ; VBE_DISPI_INDEX_YRES
10990 00003F12 66EF <1> out dx, ax
10991 <1> ;mov dx, 1CFh ; VBE_DISPI_IOPORT_DATA
10992 <1> ; 03/08/2022
10993 00003F14 FEC2 <1> inc dl
10994 00003F16 66ED <1> in ax, dx
10995 00003F18 50 <1> push eax
10996 00003F19 66BAD403 <1> mov dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
10997 00003F1D 88C4 <1> mov ah, al
10998 00003F1F B012 <1> mov al, 12h ; Vertical display end register
10999 00003F21 66EF <1> out dx, ax
11000 00003F23 58 <1> pop eax
11001 00003F24 B007 <1> mov al, 07h ; Overflow register
11002 00003F26 EE <1> out dx, al
11003 <1> ;inc dx
11004 <1> ; 03/08/2022
11005 00003F27 FEC2 <1> inc dl
11006 00003F29 EC <1> in al, dx ; read overflow register
11007 00003F2A 24BD <1> and al, 0BDh ; clear VDE 9th and 10th bits
11008 00003F2C F6C401 <1> test ah, 01h
11009 00003F2F 7402 <1> jz short bit8_clear
11010 00003F31 0C02 <1> or al, 02h ; VDE 9th bit (bit 8) in bit 1
11011 <1> bit8_clear:
11012 00003F33 F6C402 <1> test ah, 02h
11013 00003F36 7402 <1> jz short bit9_clear
11014 00003F38 0C40 <1> or al, 40h ; VDE 10th bit (bit 9) in bit 6
11015 <1> bit9_clear:
11016 00003F3A EE <1> out dx, al
11017 <1>
11018 <1> ; other settings
11019 <1> ;mov dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
11020 <1> ; 03/08/2022
11021 00003F3B B2D4 <1> mov dl, 0D4h
11022 00003F3D 66B80900 <1> mov ax, 0009h ; Maximum scan line register
11023 00003F41 66EF <1> out dx, ax ; Reset
11024 00003F43 B017 <1> mov al, 17h ; Mode control register
11025 00003F45 EE <1> out dx, al
11026 <1> ;mov dx, 3D5h ; VGAREG_VGA_CRTC_DATA
11027 00003F46 FEC2 <1> inc dl
11028 00003F48 EC <1> in al, dx ; Read mode control register
11029 00003F49 0C03 <1> or al, 03h ; Set SRS and CMS bits
11030 00003F4B EE <1> out dx, al
11031 <1> ;mov dx, 3DAh ; VGAREG_ACTL_RESET
11032 <1> ; 03/08/2022
11033 00003F4C B2DA <1> mov dl, 0DAh
11034 00003F4E EC <1> in al, dx ; clear flip-flop
11035 <1> ;mov dx, 3C0h ; VGAREG_ACTL_ADDRESS
11036 <1> ; 03/08/2022
11037 00003F4F B2C0 <1> mov dl, 0C0h
11038 00003F51 B010 <1> mov al, 10h ; Mode control register
11039 00003F53 EE <1> out dx, al
11040 <1> ;mov dx, 3C1h ; VGAREG_ACTL_READ_DATA
11041 00003F54 FEC2 <1> inc dl
11042 00003F56 EC <1> in al, dx
11043 00003F57 0C01 <1> or al, 01h ; select graphics mode
11044 <1> ;mov dx, 3C0h ; VGAREG_ACTL_ADDRESS
11045 00003F59 FECA <1> dec dl
11046 00003F5B EE <1> out dx, al ; write to mode control register
11047 00003F5C B020 <1> mov al, 20h ; Palette RAM <-> display memory
11048 00003F5E EE <1> out dx, al ; write to attribute addr register
11049 <1> ;mov dx, 3CEh ; VGAREG_GRDC_ADDRESS
11050 <1> ; 03/08/2022
11051 00003F5F B2CE <1> mov dl, 0CEh
11052 00003F61 66B80605 <1> mov ax, 0506h ; Misc. register, graph, mm 1
11053 00003F65 66EF <1> out dx, ax
11054 <1> ;mov dx, 3C4h ; VGAREG_SEQU_ADDRESS
11055 <1> ; 03/08/2022
11056 00003F67 B2C4 <1> mov dl, 0C4h
11057 00003F69 66B8020F <1> mov ax, 0F02h ; Map mask register, all planes
11058 00003F6D 66EF <1> out dx, ax
11059 <1>
11060 <1> ; settings for >= 8bpp
11061 <1>
11062 <1> ;mov dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
11063 <1> ;mov ax, 03h ; VBE_DISPI_INDEX_BPP
11064 <1> ;out dx, ax
11065 <1> ;;mov dx, 1CFh ; VBE_DISPI_IOPORT_DATA
11066 <1> ;inc dl
11067 <1> ;in ax, dx
11068 <1> ;cmp al, 08h ; < 8 bits per pixel
11069 <1> ;jb short vga_compat_end
11070 <1>
11071 <1> ;mov dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
11072 <1> ; 03/08/2022
11073 00003F6F B2D4 <1> mov dl, 0D4h
11074 00003F71 B014 <1> mov al, 14h ; Underline location register
11075 00003F73 EE <1> out dx, al
11076 <1> ;mov dx, 3D5h ; VGAREG_VGA_CRTC_DATA
11077 00003F74 FEC2 <1> inc dl
11078 00003F76 EC <1> in al, dx
11079 00003F77 0C40 <1> or al, 40h ; enable double word mode
11080 00003F79 EE <1> out dx, al

```

```

11081      <1>      ;mov     dx, 3DAh ; VGAREG_ACTL_RESET
11082      <1>      ; 03/08/2022
11083      00003F7A B2DA      <1>      mov     dl, 0DAh
11084      00003F7C EC        <1>      in      al, dx ; clear flip-flop
11085      <1>      ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
11086      <1>      ; 03/08/2022
11087      00003F7D B2C0      <1>      mov     dl, 0C0h
11088      00003F7F B010      <1>      mov     al, 10h ; Mode control register
11089      00003F81 EE        <1>      out     dx, al
11090      <1>      ;mov     dx, 3C1h ; VGAREG_ACTL_READ_DATA
11091      00003F82 FEC2      <1>      inc     dl
11092      00003F84 EC        <1>      in      al, dx
11093      00003F85 0C40      <1>      or      al, 40h ; Pixel clock select is 1
11094      <1>      ;mov     dx, 3C0h ; VGAREG_ACTL_ADDRESS
11095      00003F87 FECA      <1>      dec     dl
11096      00003F89 EE        <1>      out     dx, al ; update mode control reggister
11097      00003F8A B020      <1>      mov     al, 20h ; select display memory as PAS
11098      00003F8C EE        <1>      out     dx, al
11099      <1>      ;mov     dx, 3C4h ; VGAREG_SEQU_ADDRESS
11100      <1>      ; 03/08/2022
11101      00003F8D B2C4      <1>      mov     dl, 0C4h
11102      00003F8F B004      <1>      mov     al, 04h ; Memory mode register
11103      00003F91 EE        <1>      out     dx, al
11104      <1>      ;mov     dx, 3C5h ; VGAREG_SEQU_DATA
11105      00003F92 FEC2      <1>      inc     dl
11106      00003F94 EC        <1>      in      al, dx
11107      00003F95 0C08      <1>      or      al, 08h ; enable chain four
11108      00003F97 EE        <1>      out     dx, al
11109      <1>      ;mov     dx, 3CEh ; VGAREG_GRDC_ADDRESS
11110      <1>      ; 03/08/2022
11111      00003F98 B2CE      <1>      mov     dl, 0CEh
11112      00003F9A B005      <1>      mov     al, 05h ; Mode register
11113      00003F9C EE        <1>      out     dx, al
11114      <1>      ;mov     dx, 3CFh ; VGAREG_GRDC_DATA
11115      00003F9D FEC2      <1>      inc     dl
11116      00003F9F EC        <1>      in      al, dx
11117      00003FA0 249F      <1>      and     al, 9Fh ; clear shift register
11118      00003FA2 0C40      <1>      or      al, 40h ; set shift register to 2
11119      00003FA4 EE        <1>      out     dx, al
11120      <1>
11121      <1> vga_compat_end:
11122      <1>      ;pop     edx
11123      <1>      ;pop     eax
11124      00003FA5 C3        <1>      retn
11125      <1>
11126      <1> vga_set_virt_width:
11127      <1>      ; 03/08/2022
11128      <1>      ; 27/11/2020
11129      <1>      ; 25/11/2020
11130      <1>      ; (vbe.c, 02/01/2020, vruppert)
11131      <1>      ;
11132      <1>      ; Input:
11133      <1>      ;      AX = resolution (screen width)
11134      <1>      ;
11135      <1>      ; Modified registers: eax, edx
11136      <1>
11137      <1>      ;;push ebx
11138      <1>      ;push edx
11139      <1>      ;push eax
11140      <1>      ;mov ebx, eax
11141      <1>      ;call dispi_get_bpp ; bits per pixel
11142      <1>      ;cmp al, 4
11143      <1>      ;ja short set_width_svgaga ; 8, 16, 24, 32
11144      <1>      ;shr bx, 1
11145      <1> ;set_width_svgaga:
11146      <1>      ;shr bx, 3
11147      <1>      ;mov eax, [esp]
11148      <1>      ;shr ax, 3 ; / 8, bytes per row
11149      <1>      ; 03/08/2022
11150      00003FA6 C1E803      <1>      shr     eax, 3
11151      00003FA9 66BAD403    <1>      mov     dx, 3D4h ; VGAREG_VGA_CRTC_ADDRESS
11152      <1>      ;mov     ah, bl ;
11153      00003FAD 88C4        <1>      mov     ah, al ; width in bytes
11154      00003FAF B013        <1>      mov     al, 13h ; offset register
11155      00003FB1 66EF        <1>      out     dx, ax ; index (3D4h) and data (3D5h)
11156      <1>      ;pop     eax
11157      <1>      ;pop     edx
11158      <1>      ;pop     ebx
11159      00003FB3 C3        <1>      retn
11160      <1>
11161      <1> ; 24/11/2020
11162      <1>
11163      <1> struc bmi ; BOCHS/PLEX86 MODE INFO structure/table
11164      00000000 ????      <1>      .mode:      resw 1
11165      00000002 ????      <1>      .width:     resw 1
11166      00000004 ????      <1>      .height:    resw 1
11167      00000006 ????      <1>      .depth:     resw 1
11168      <1>      .size:
11169      <1>      endstruc
11170      <1>
11171      <1> ; 24/11/2020
11172      <1> struc MODEINFO
11173      00000000 ????      <1>      .mode:      resw 1 ; 1xxh
11174      00000002 ????      <1>      .ModeAttributes: resw 1
11175      00000004 ??        <1>      .WinAttributes: resb 1
11176      00000005 ??        <1>      .WinBAttributes: resb 1 ; = 0
11177      00000006 ????      <1>      .WinGranularity: resw 1
11178      00000008 ????      <1>      .WinSize:      resw 1
11179      0000000A ????      <1>      .WinASegment:   resw 1
11180      0000000C ????      <1>      .WinBSegment:   resw 1 ; = 0
11181      0000000E ????      <1>      .WinFuncPtr:    resd 1 ; = 0
11182      00000012 ????      <1>      .BytesPerScanLine: resw 1
11183      00000014 ????      <1>      .XResolution:    resw 1
11184      00000016 ????      <1>      .YResolution:    resw 1
11185      00000018 ??        <1>      .XCharSize:      resb 1
11186      00000019 ??        <1>      .YCharSize:      resb 1
11187      0000001A ??        <1>      .NumberOfPlanes: resb 1
11188      0000001B ??        <1>      .BitsPerPixel:   resb 1
11189      0000001C ??        <1>      .NumberOfBanks:  resb 1
11190      0000001D ??        <1>      .MemoryModel:    resb 1
11191      0000001E ??        <1>      .BankSize:       resb 1 ; = 0
11192      0000001F ??        <1>      .NumberOfImagePages: resb 1
11193      00000020 ??        <1>      .Reserved_page:  resb 1 ; = 0
11194      00000021 ??        <1>      .RedMaskSize:    resb 1
11195      00000022 ??        <1>      .RedFieldPosition: resb 1
11196      00000023 ??        <1>      .GreenMaskSize:   resb 1
11197      00000024 ??        <1>      .GreenFieldPosition: resb 1
11198      00000025 ??        <1>      .BlueMaskSize:    resb 1
11199      00000026 ??        <1>      .BlueFieldPosition: resb 1
11200      00000027 ??        <1>      .RsvdMaskSize:    resb 1
11201      00000028 ??        <1>      .RsvdFieldPosition: resb 1
11202      00000029 ??        <1>      .DirectColorModeInfo: resb 1
11203      0000002A ????      <1>      .PhysBasePtr:    resd 1
11204      0000002E ????      <1>      .OffScreenMemOffset: resd 1 ; = 0

```

```

11205 00000032 ????.<1> .OffScreenMemSize: resw 1 ; = 0
11206 00000034 ????.<1> .LinBytesPerScanLine: resw 1
11207 00000036 ??<1> .BnkNumberOfPages: resb 1
11208 00000037 ??<1> .LinNumberOfPages: resb 1
11209 00000038 ??<1> .LinRedMaskSize: resb 1
11210 00000039 ??<1> .LinRedFieldPosition1: resb 1
11211 0000003A ??<1> .LinGreenMaskSize: resb 1
11212 0000003B ??<1> .LinGreenFieldPosition: resb 1
11213 0000003C ??<1> .LinBlueMaskSize: resb 1
11214 0000003D ??<1> .LinBlueFieldPosition: resb 1
11215 0000003E ??<1> .LinRsvdMaskSize: resb 1
11216 0000003F ??<1> .LinRsvdFieldPosition: resb 1
11217 00000040 ?????????<1> .MaxPixelClock: resd 1 ; = 0
11218<1> .size:
11219<1> endstruc
11220<1>
11221<1> ; 10/12/2020
11222<1> struc LFBINFO
11223 00000000 ????.<1> .mode: resw 1 ; 1XXh
11224 00000002 ?????????<1> .LFB_addr: resd 1
11225 00000006 ?????????<1> .LFB_size: resd 1
11226 0000000A ????.<1> .X_res: resw 1
11227 0000000C ????.<1> .Y_res: resw 1
11228 0000000E ??<1> .bpp: resb 1
11229 0000000F ??<1> .reserved: resb 1
11230<1> .size: ; 16 bytes
11231<1> endstruc
11232<1>
11233<1> set_mode_info_list:
11234<1> ; 14/12/2020
11235<1> ; 11/12/2020
11236<1> ; 24/11/2020
11237<1> ; (vbetables-gen.c)
11238<1> ; Input:
11239<1> ; BX = VBE mode (including bochs special modes)
11240<1> ; Output:
11241<1> ; ;EAX = MODE_INFO_LIST address
11242<1> ; EAX = 0 ; 11/12/2020
11243<1> ; ESI = MODE_INFO_LIST address ; 11/12/2020
11244<1> ; (if mode is not found, ESI = 0)
11245<1> ;
11246<1> ; Modified registers: eax, ebx, ecx, edx, esi, edi
11247<1>
11248 00003FB4 BE[AA6B0000]<1> mov esi, b_vbe_modes ; bochs mode info base table
11249 00003FB9 BF[2A9D0100]<1> mov edi, MODE_INFO_LIST ; mode info list (4F01h)
11250<1> sml_0:
11251 00003FBE 66AD<1> lodsw
11252 00003FC0 6639D8<1> cmp ax, bx ; is mode number same ?
11253 00003FC3 7410<1> je short sml_1 ; yes
11254 00003FC5 AD<1> lodsd
11255 00003FC6 66AD<1> lodsw
11256 00003FC8 81FE[6A6C0000]<1> cmp esi, end_of_b_vbe_modes
11257 00003FCE 72EE<1> jb short sml_0
11258<1> ; not found
11259 00003FD0 31C0<1> xor eax, eax ; 0
11260<1> ; 11/12/2020
11261 00003FD2 31F6<1> xor esi, esi
11262 00003FD4 C3<1> retn
11263<1> sml_1:
11264 00003FD5 66AB<1> stosw ; mode
11265 00003FD7 AD<1> lodsd ; width, height
11266<1> ; 14/12/2020
11267 00003FD8 89C1<1> mov ecx, eax
11268 00003FDA 50<1> push eax ; ***
11269 00003FDB 29C0<1> sub eax, eax ; clear high word of eax
11270 00003FDD 66AD<1> lodsw ; depth
11271 00003FDF 50<1> push eax ; **
11272<1>
11273<1> ;add al, 7 ; only for 15 bit colors (not used here)
11274 00003FE0 C0E803<1> shr al, 3 ; / 8
11275<1> ; 14/12/2020
11276 00003FE3 66F7E1<1> mul cx ; pitch = width * ((depth+7)/8)
11277<1> ; ax = pitch
11278 00003FE6 50<1> push eax ; * ; high word of eax = 0
11279 00003FE7 C1E910<1> shr ecx, 16
11280<1> ;mul cx
11281<1> ;mov cx, ax
11282 00003FEA 31D2<1> xor edx, edx ; clear high word of edx
11283 00003FEC F7E1<1> mul ecx ; height * pitch
11284 00003FEE 89C1<1> mov ecx, eax
11285 00003FF0 B800000001<1> mov eax, VBE_DISPI_TOTAL_VIDEO_MEMORY_MB * 1024 * 1024
11286 00003FF5 F7F1<1> div ecx
11287<1> ; eax = pages = vram_size / (height*pitch)
11288<1>
11289<1> ;mov cx, ax
11290 00003FF7 89C1<1> mov ecx, eax ; pages
11291<1>
11292 00003FF9 66B89B00<1> mov ax, MODE_ATTRIBUTES
11293 00003FFD 66AB<1> stosw ; ModeAttributes
11294 00003FFF B007<1> mov al, WINA_ATTRIBUTES
11295 00004001 AA<1> stosb ; winAAttributes
11296 00004002 30C0<1> xor al, al ; winBAttributes = 0
11297 00004004 AA<1> stosb
11298 00004005 66B84000<1> mov ax, VBE_DISPI_BANK_SIZE_KB
11299 00004009 66AB<1> stosw ; winGranularity
11300 0000400B 66AB<1> stosw ; winSize
11301 0000400D 66B800A0<1> mov ax, VGAMEM_GRAPH
11302 00004011 66AB<1> stosw ; winASegment
11303 00004013 29C0<1> sub eax, eax
11304 00004015 66AB<1> stosw ; winBSegment = 0
11305 00004017 AB<1> stosd ; winFuncPtr = 0
11306<1>
11307 00004018 58<1> pop eax ; * ; pitch
11308 00004019 89C3<1> mov ebx, eax ; high word of ebx = 0 ; 14/12/2020
11309 0000401B 66AB<1> stosw ; BytesPerScanLine
11310<1>
11311 0000401D 5A<1> pop edx ; ** ; depth (bits per pixel)
11312 0000401E 58<1> pop eax ; *** width, height
11313<1>
11314<1> ; // Mandatory information for VBE 1.2 and above
11315<1>
11316 0000401F 66AB<1> stosw ; XResolution (width)
11317 00004021 C1E810<1> shr eax, 16
11318 00004024 50<1> push eax ; **** height
11319 00004025 66AB<1> stosw ; YResolution (height)
11320 00004027 B008<1> mov al, 8
11321 00004029 AA<1> stosb ; XCharSize ; char width
11322 0000402A B010<1> mov al, 16
11323 0000402C AA<1> stosb ; YCharSize ; char height
11324 0000402D B001<1> mov al, 1
11325 0000402F AA<1> stosb ; NumberOfPlanes
11326<1> ;movzx eax, dl
11327 00004030 88D0<1> mov al, dl ; eax <= 32
11328 00004032 AA<1> stosb ; BitsPerPixel

```

```

11329          <1>      ; Number of banks = (height * pitch + 65535) / 65536
11330 00004033 58      <1>      pop     eax ; ***** ; height
11331          <1>      ; 14/12/2020
11332 00004034 52      <1>      push    edx ; ***** ; depth ; edx <= 32
11333 00004035 F7E3    <1>      mul     ebx ; pitch (ebx) * height (eax)
11334          <1>      ;mov    edx, [esp] ; *****
11335          <1>      ;mov    dl, [esp] ; *****
11336 00004037 05FFFF0000 <1>      add     eax, 65535
11337 0000403C C1E810  <1>      shr     eax, 16 ; / 65536 ; <= 127 ; 14/12/2020
11338 0000403F AA      <1>      stosb   ; NumberOfBanks
11339          <1>      ; 14/12/2020
11340          <1>      ;cmp     dl, 8 ; 8 bits per pixel
11341 00004040 803C2408 <1>      cmp     byte [esp], 8
11342 00004044 7704    <1>      ja      short sm1_2
11343 00004046 B004    <1>      mov     al, VBE_MEMORYMODEL_PACKED_PIXEL
11344 00004048 EB02    <1>      jmp     short sm1_3
11345          <1>      sm1_2:
11346          <1>      ; 16, 24, 32 bits per pixel
11347 0000404A B006    <1>      mov     al, VBE_MEMORYMODEL_DIRECT_COLOR
11348          <1>      sm1_3:
11349 0000404C AA      <1>      stosb   ; BankSize = 0
11350 0000404D 30C0    <1>      xor     al, al ; 0
11351 0000404F AA      <1>      stosb   ; BankSize = 0
11352 00004050 49      <1>      dec     ecx ; pages - 1
11353          <1>      ; NumberOfImagePages = 262 for 320x200x8 mode
11354          <1>      ;mov     ax, 255
11355          <1>      ; 14/12/2020
11356          <1>      ;mov     al, 255
11357 00004051 FEC8    <1>      dec     al ; 255
11358 00004053 39C1    <1>      cmp     ecx, eax ; ecx <= 261, eax = 255
11359          <1>      ;cmp     cx, ax
11360 00004055 7302    <1>      jnb     short sm1_4
11361 00004057 88C8    <1>      mov     al, cl
11362          <1>      sm1_4:
11363 00004059 AA      <1>      stosb   ; NumberOfImagePages (1 byte)
11364 0000405A 28C0    <1>      sub     al, al
11365 0000405C AA      <1>      stosb   ; Reserved_page = 0
11366 0000405D 58      <1>      pop     eax ; ***** ; depth
11367 0000405E 88C1    <1>      mov     cl, al
11368          <1>      ; eax <= 32
11369 00004060 2C08    <1>      sub     al, 8 ; 8->0, 16->8, 24->16, 32->24
11370 00004062 BE[6A6C0000] <1>      mov     esi, direct_color_fields
11371 00004067 01C6    <1>      add     esi, eax
11372 00004069 56      <1>      push    esi ; *****
11373 0000406A AD      <1>      lodsd   ; RedMaskSize (AL), RedFieldPosition (AH)
11374          <1>      ; GreenMaskSize (16), GreenFieldPosition (24)
11375 0000406B AB      <1>      stosd   ; BlueMaskSize (AL), BlueFieldPosition (AH)
11376 0000406C AD      <1>      lodsd   ; RsvdMaskSize (16), RsvdFieldPosition (24)
11377          <1>      ; RsvdMaskSize (16), RsvdFieldPosition (24)
11378 0000406D AB      <1>      stosd   ; BlueMaskSize (AL), BlueFieldPosition (AH)
11379 0000406E 5E      <1>      pop     esi ; *****
11380          <1>      ; *****
11381 0000406F 30C0    <1>      xor     al, al ; 0
11382 00004071 80F920  <1>      cmp     cl, 32
11383 00004074 7202    <1>      jb      short sm1_5
11384 00004076 B002    <1>      mov     al, VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
11385          <1>      sm1_5:
11386 00004078 AA      <1>      stosb   ; DirectColorModeInfo
11387          <1>      ; *****
11388          <1>      ; // Mandatory information for VBE 2.0 and above
11389          <1>      ; *****
11390 00004079 B8000000E0 <1>      mov     eax, VBE_DISPI_LFB_PHYSICAL_ADDRESS
11391 0000407E AB      <1>      stosd   ; PhysBasePtr
11392 0000407F 29C0    <1>      sub     eax, eax
11393 00004081 AB      <1>      stosd   ; OffScreenMemOffset = 0
11394 00004082 66AB    <1>      stosw   ; OffScreenMemSize = 0
11395          <1>      ; *****
11396          <1>      ;// Mandatory information for VBE 3.0 and above
11397          <1>      ; *****
11398          <1>      ; ebx = pitch
11399 00004084 6689D8  <1>      mov     ax, bx
11400          <1>      ;stosw
11401          <1>      ; *****
11402          <1>      ;xor     al, al
11403          <1>      ;stosb ; BnkNumberOfPages = 0
11404          <1>      ;stosb ; LinNumberOfPages = 0
11405          <1>      ; *****
11406 00004087 AB      <1>      stosd   ; pitch (word), 0 (byte), 0 (byte)
11407          <1>      ; *****
11408 00004088 AD      <1>      lodsd   ; LinRedMaskSize (AL), LinRedFieldPosition (AH)
11409          <1>      ; LinGreenMaskSize (16), LinGreenFieldPosition (24)
11410 00004089 AB      <1>      stosd   ; LinBlueMaskSize (AL), LinBlueFieldPosition (AH)
11411 0000408A AD      <1>      lodsd   ; LinRsvdMaskSize (16), LinRsvdFieldPosition (24)
11412          <1>      ; LinRsvdMaskSize (16), LinRsvdFieldPosition (24)
11413 0000408B AB      <1>      stosd   ; LinBlueMaskSize (AL), LinBlueFieldPosition (AH)
11414          <1>      ; LinRsvdMaskSize (16), LinRsvdFieldPosition (24)
11415 0000408C 29C0    <1>      sub     eax, eax
11416 0000408E AB      <1>      stosd   ; MaxPixelClock = 0
11417          <1>      ; *****
11418          <1>      ;mov     eax, MODE_INFO_LIST
11419          <1>      ; 11/12/2020
11420 0000408F BE[2A9D0100] <1>      mov     esi, MODE_INFO_LIST
11421          <1>      ; *****
11422 00004094 C3      <1>      retn
11423          <1>      ; *****
11424          <1>      ; end of set_mode_info_list ; edi = set_mode_info_list + 68
11425          <1>      ; *****
11426          <1>      pci_get_lfb_addr:
11427          <1>      ; 11/12/2020
11428          <1>      ; Get linear frame buffer base from PCI
11429          <1>      ; (vgabios.c, 02/01/2020, vruppert)
11430          <1>      ; *****
11431          <1>      ; Input:
11432          <1>      ; ax = PCI device vendor id
11433          <1>      ; Output:
11434          <1>      ; ax = LFB address (high 16 bit) (zf=0)
11435          <1>      ; eax = 0 -> not found (error) (zf=1)
11436          <1>      ; *****
11437          <1>      ; Modified registers: eax
11438          <1>      ; *****
11439 00004095 53      <1>      push    ebx
11440 00004096 51      <1>      push    ecx
11441 00004097 52      <1>      push    edx
11442          <1>      ; *****
11443 00004098 89C3    <1>      mov     ebx, eax
11444 0000409A 31C9    <1>      xor     ecx, ecx
11445 0000409C 28D2    <1>      sub     dl, dl ; mov dl, 0
11446 0000409E E842000000 <1>      call    pci_read_reg
11447 000040A3 6683F8FF <1>      cmp     ax, 0FFFFh
11448 000040A7 7417    <1>      je      short pci_get_lfb_addr_fail
11449          <1>      pci_get_lfb_addr_next_dev:
11450 000040A9 28D2    <1>      sub     dl, dl ; mov dl, 0
11451 000040AB E835000000 <1>      call    pci_read_reg
11452 000040B0 6639D8  <1>      cmp     ax, bx ; check vendor

```

```

11453 000040B3 740F      <1>      je      short pci_get_lfb_addr_found
11454 000040B5 6683C108    <1>      add     cx, 08h
11455 000040B9 6681F90002    <1>      cmp     cx, 200h ; search bus 0 and 1
11456 000040BE 72E9      <1>      jb      short pci_get_lfb_addr_next_dev
11457                                <1> pci_get_lfb_addr_fail:
11458 000040C0 31C0      <1>      xor     eax, eax ; no LFB
11459                                <1>      ; zf = 1
11460 000040C2 EB1D      <1>      jmp     short pci_get_lfb_addr_return
11461                                <1> pci_get_lfb_addr_found:
11462 000040C4 B210      <1>      mov     dl, 10h ; I/O space 0
11463 000040C6 E81A000000    <1>      call    pci_read_reg
11464 000040CB 66A9F1FF    <1>      test    ax, 0FFFF1h
11465 000040CF 740D      <1>      jz      short pci_get_lfb_addr_success
11466 000040D1 B214      <1>      mov     dl, 14h ; I/O space 1
11467 000040D3 E80D000000    <1>      call    pci_read_reg
11468 000040D8 66A9F1FF    <1>      test    ax, 0FFFF1h
11469 000040DC 75E2      <1>      jnz     short pci_get_lfb_addr_fail
11470                                <1> pci_get_lfb_addr_success:
11471 000040DE C1E810    <1>      shr     eax, 16 ; LFB address (hw)
11472                                <1>      ; zf = 0
11473                                <1> pci_get_lfb_addr_return:
11474 000040E1 5A      <1>      pop     edx
11475 000040E2 59      <1>      pop     ecx
11476 000040E3 5B      <1>      pop     ebx
11477 000040E4 C3      <1>      retn
11478                                <1>
11479                                <1> pci_read_reg:
11480                                <1>      ; 11/12/2020
11481                                <1>      ; Read PCI register
11482                                <1>      ; (vgabios.c, 02/01/2020, vruppert)
11483                                <1>      ;
11484                                <1>      ; Input:
11485                                <1>      ;     cx = device/function
11486                                <1>      ;     dl = register
11487                                <1>      ; Output:
11488                                <1>      ;     eax = value
11489                                <1>      ;
11490                                <1>      ; Modified registers: eax, edx
11491                                <1>
11492 000040E5 B800008000    <1>      mov     eax, 00800000h
11493 000040EA 6689C8      <1>      mov     ax, cx
11494 000040ED C1E008      <1>      shl     eax, 8
11495 000040F0 88D0      <1>      mov     al, dl
11496 000040F2 66BAF80C    <1>      mov     dx, 0CF8h
11497 000040F6 EF      <1>      out     dx, eax
11498 000040F7 80C204      <1>      add     dl, 4 ; mov dx, 0CFCh
11499 000040FA ED      <1>      in      eax, dx
11500 000040FB C3      <1>      retn
11501                                <1>
11502                                <1> %endif
11503                                <1>
11504                                <1> ; -----
11505                                <1>
11506                                <1> %if 0
11507                                <1>
11508                                <1> mode_info_find_mode:
11509                                <1>      ; 25/11/2020
11510                                <1>      ; Input:
11511                                <1>      ;     bx = VESA VBE2 video mode (+ bochs extensions)
11512                                <1>      ; Output:
11513                                <1>      ;     esi = mode info address (for BX input)
11514                                <1>      ;     esi = 0 -> not found
11515                                <1>      ;
11516                                <1>      ; Modified registers: eax, esi
11517                                <1>
11518                                <1>      xor     eax, eax
11519                                <1>      mov     esi, MODE_INFO_LIST
11520                                <1> mifm_1:
11521                                <1>      mov     ax, [esi]
11522                                <1>      cmp     ax, bx
11523                                <1>      je      short mifm_2
11524                                <1>      add     esi, MODEINFO.size ; add esi, 68
11525                                <1>      cmp     esi, VBE_VESA_MODE_END_OF_LIST
11526                                <1>      jb      short mifm_1
11527                                <1>      ; not found
11528                                <1>      sub     esi, esi ; 0
11529                                <1> mifm_2
11530                                <1>      retn
11531                                <1>
11532                                <1> dispi_get_bpp:
11533                                <1>      ; 28/11/2020
11534                                <1>      ; Input:
11535                                <1>      ;     none
11536                                <1>      ; Output:
11537                                <1>      ;     al = Bits per pixel
11538                                <1>      ;     (8,16,24,32)
11539                                <1>      ;     ah = Bytes per pixel
11540                                <1>      ;     (1,2,3,4)
11541                                <1>      ;
11542                                <1>      ; Modified registers: none
11543                                <1>
11544                                <1>      ;push    edx
11545                                <1>      ;mov     dx, 01CEh ; VBE_DISPI_IOPORT_INDEX
11546                                <1>      ;mov     ax, 03h ; VBE_DISPI_INDEX_BPP
11547                                <1>      ;out     dx, ax
11548                                <1>      ;;mov    dx, 01CFh ; VBE_DISPI_IOPORT_DATA
11549                                <1>      ;;mov    dl, 0CFh
11550                                <1>      ;inc     dl
11551                                <1>      ;in      ax, dx
11552                                <1>      ;mov     ah, al
11553                                <1>      ;shr     ah, 3 ; / 8
11554                                <1>      ;;test   al, 7 ; 15 bit graphics mode
11555                                <1>      ;;jz     short get_bpp_noinc
11556                                <1>      ;;inc     ah
11557                                <1> ;;get_bpp_noinc:
11558                                <1>      ;pop     edx
11559                                <1>      ;retn
11560                                <1>
11561                                <1>      ; 28/11/2020
11562                                <1>      ; Modified registers: edx
11563                                <1>      ;push    edx
11564                                <1>      ;mov     dx, 03h ; VBE_DISPI_INDEX_BPP
11565                                <1>      ;call    dispi_get_parms
11566                                <1>      ;pop     edx
11567                                <1>      ;retn
11568                                <1>      ;mov     ah, al
11569                                <1>      ;shr     ah, 3 ; / 8
11570                                <1>      ;test   al, 7 ; 15 bit graphics mode
11571                                <1>      ;jz     short get_bpp_noinc
11572                                <1>      ;inc     ah
11573                                <1> ;get_bpp_noinc:
11574                                <1>      ;pop     edx
11575                                <1>      ;retn
11576                                <1>

```

```

11577 <1> restore_vesa_video_state:
11578 <1> ; 02/08/2022 (TRDOS 386 v2.0.5)
11579 <1> ; 14/01/2021
11580 <1> ; 06/12/2020
11581 <1> ; Input:
11582 <1> ; [vbe3stbufsize] <= 32 ; <= 32*64 bytes
11583 <1> ; Output:
11584 <1> ; AX = 004Fh (succeeded)
11585 <1> ;
11586 <1> ; eax = 0 -> buffer size problem
11587 <1> ; eax > 0 and ax <> 004Fh -> failed
11588 <1> ;
11589 <1> ; Modified regs: eax, ebx, ecx, edx, esi, edi
11590 <1>
11591 <1> ;movzx ecx, word [vbe3stbufsize]
11592 <1> ;cmp cx, 32 ; 32 * 64 bytes
11593 <1> ;ja short r_v_b_s_fail
11594 <1>
11595 <1> movzx ecx, byte [vbe3stbufsize]; <=32
11596 <1> ;shl cx, 4 ; dword count for movsd
11597 <1> ; 02/08/2022
11598 <1> shl ecx, 4 ; <= 32*16 dwords (32*64 bytes)
11599 <1> mov edi, VBE3SAVERESTOREBLOCK ; destination
11600 <1> ; (vbe3 pmi buff)
11601 <1> mov esi, VBE3VIDEOSTATE ; source (kernel buff)
11602 <1> rep movsd
11603 <1>
11604 <1> mov ax, 4F04h
11605 <1> mov dl, 02h ; restore
11606 <1> ;mov cx, 0Fh
11607 <1> mov cl, 0Fh
11608 <1> xor ebx, ebx ; points to VBE3SAVERESTOREBLOCK
11609 <1> jmp short int10h_32bit_pmi
11610 <1>
11611 <1> ;s_v_b_s_fail:
11612 <1> ;r_v_b_s_fail:
11613 <1> ; xor eax, eax
11614 <1> ; retn
11615 <1>
11616 <1> save_vesa_video_state:
11617 <1> ; 02/08/2022 (TRDOS 386 v2.0.5)
11618 <1> ; 14/01/2021
11619 <1> ; 06/12/2020
11620 <1> ; Input:
11621 <1> ; [vbe3stbufsize] <= 32 ; <= 32*64 bytes
11622 <1> ; Output:
11623 <1> ; AX = 004Fh (succeeded)
11624 <1> ;
11625 <1> ; eax = 0 -> buffer size problem
11626 <1> ; eax > 0 and ax <> 004Fh -> failed
11627 <1> ;
11628 <1> ; Modified regs: eax, ebx, ecx, edx, esi, edi
11629 <1>
11630 <1> ;cmp word [vbe3stbufsize], 32
11631 <1> ; ; 32 * 64 bytes
11632 <1> ;ja short s_v_b_s_fail
11633 <1>
11634 <1> mov ax, 4F04h
11635 <1> mov dl, 01h ; save
11636 <1> ;mov cx, 0Fh
11637 <1> mov cl, 0Fh
11638 <1> xor ebx, ebx ; points to VBE3SAVERESTOREBLOCK
11639 <1>
11640 <1> call int10h_32bit_pmi
11641 <1>
11642 <1> movzx ecx, byte [vbe3stbufsize]; <=32
11643 <1> ;shl cx, 4 ; dword count for movsd
11644 <1> ; 02/08/2022
11645 <1> shl ecx, 4 ; <= 32*16 dwords (32*64 bytes)
11646 <1> mov esi, VBE3SAVERESTOREBLOCK ; destination
11647 <1> ; (vbe3 pmi buff)
11648 <1> mov edi, VBE3VIDEOSTATE ; source (kernel buff)
11649 <1> rep movsd
11650 <1> retn
11651 <1>
11652 <1> dispi_set_bank_farcall:
11653 <1> ; 03/08/2022
11654 <1> ; 12/04/2021 (32 bit push/pop)
11655 <1> ; 11/12/2020
11656 <1> ; (This may be 'sysvideo' function, later)
11657 <1> ;
11658 <1> ; Input:
11659 <1> ; bx = 0000h, set bank number
11660 <1> ; = 0100h, get bank number
11661 <1> ; dx = bank number (if bx = 0)
11662 <1> ; Output:
11663 <1> ; dx = bank number
11664 <1>
11665 <1> cmp bx, 0100h
11666 <1> je short dispi_set_bank_farcall_get
11667 <1> or bx, bx
11668 <1> ;jnz dispi_set_bank_farcall_error
11669 <1> ; 03/08/2022
11670 <1> jz short dsbfcall_1
11671 <1> jmp dispi_set_bank_farcall_error
11672 <1> dsbfcall_1:
11673 <1> mov ax, dx
11674 <1> ;push dx
11675 <1> ;push ax
11676 <1> ; 12/04/2021
11677 <1> push edx
11678 <1> push eax
11679 <1> mov ax, VBE_DISPI_INDEX_BANK
11680 <1> mov dx, 1CEh ; VBE_DISPI_IOPORT_INDEX
11681 <1> out dx, ax
11682 <1> ;pop ax
11683 <1> ; 12/04/2021
11684 <1> pop eax
11685 <1> ;mov dx, VBE_DISPI_IOPORT_DATA
11686 <1> ;mov dl, 0CFh
11687 <1> inc dl ; 1CFh = VBE_DISPI_IOPORT_DATA
11688 <1> out dx, ax
11689 <1> in ax, dx
11690 <1> ;pop dx
11691 <1> ; 12/04/2021
11692 <1> pop edx
11693 <1> cmp dx, ax
11694 <1> jne short dispi_set_bank_farcall_error
11695 <1> mov ax, 004Fh
11696 <1> retn ; retf for real mode far call
11697 <1> dispi_set_bank_farcall_get:
11698 <1> mov ax, VBE_DISPI_INDEX_BANK
11699 <1> ; 03/08/2022
11700 <1> mov dx, 1CEh ; VBE_DISPI_IOPORT_INDEX

```



```

11701 <1> out dx, ax
11702 <1> ;mov dx, VBE_DISPI_IOPORT_DATA
11703 <1> ;mov dl, 0CFh
11704 <1> ; 03/08/2022
11705 <1> inc dl ; 1CFh = VBE_DISPI_IOPORT_DATA
11706 <1> in ax, dx
11707 <1> mov dx, ax
11708 <1> retf ; retf for real mode far call
11709 <1> dispi_set_bank_farcall_error:
11710 <1> mov ax, 014Fh
11711 <1> retf ; retf for real mode far call
11712 <1>
11713 <1> %endif
11714 <1>
11715 <1> ; % include 'vidata.s' ; VIDEO DATA
11716 <1>
11717 <1> ; /// End Of VIDEO FUNCTIONS ///
2991
2992 setup_rtc_int:
2993 ; source: http://wiki.osdev.org/RTC
2994 000040FC FA cli ; disable interrupts
2995 ; default int frequency is 1024 Hz (Lower 4 bits of register A is 0110b or 6)
2996 ; in order to change this ...
2997 ; frequency = 32768 >> (rate-1) --> 32768 >> 5 = 1024
2998 ; (rate must be above 2 and not over 15)
2999 ; new rate = 15 --> 32768 >> (15-1) = 2 Hz
3000 000040FD B08A mov al, 8Ah
3001 000040FF E670 out 70h, al ; set index to register A, disable NMI
3002 00004101 90 nop
3003 00004102 E471 in al, 71h ; get initial value of register A
3004 00004104 88C4 mov ah, al
3005 00004106 80E4F0 and ah, 0F0h
3006 00004109 B08A mov al, 8Ah
3007 0000410B E670 out 70h, al ; reset index to register A
3008 0000410D 88E0 mov al, ah
3009 0000410F 0C0F or al, 0Fh ; new rate (0Fh -> 15)
3010 00004111 E671 out 71h, al ; write only our rate to A. Note, rate is the bottom 4 bits.
3011 ; enable RTC interrupt
3012 00004113 B08B mov al, 8Bh ;
3013 00004115 E670 out 70h, al ; select register B and disable NMI
3014 00004117 90 nop
3015 00004118 E471 in al, 71h ; read the current value of register B
3016 0000411A 88C4 mov ah, al ;
3017 0000411C B08B mov al, 8Bh ;
3018 0000411E E670 out 70h, al ; set the index again (a read will reset the index to register B)
3019 00004120 88E0 mov al, ah ;
3020 00004122 0C40 or al, 40h ;
3021 00004124 E671 out 71h, al ; write the previous value ORed with 0x40. This turns on bit 6 of register B
3022 00004126 FB sti
3023 00004127 C3 retn
3024
3025 ; Write memory information
3026 ; 29/01/2016
3027 ; 06/11/2014
3028 ; 14/08/2015
3029 memory_info:
3030 00004128 A1[A4760100] mov eax, [memory_size] ; in pages
3031 0000412D 50 push eax
3032 0000412E C1E00C shl eax, 12 ; in bytes
3033 00004131 BB0A000000 mov ebx, 10
3034 00004136 89D9 mov ecx, ebx ; 10
3035 00004138 BE[61380100] mov esi, mem_total_b_str
3036 0000413D E8D8000000 call bintdstr
3037 00004142 58 pop eax
3038 00004143 B107 mov cl, 7
3039 00004145 BE[85380100] mov esi, mem_total_p_str
3040 0000414A E8CB000000 call bintdstr
3041 ; 14/08/2015
3042 0000414F E8E3000000 call calc_free_mem
3043 ; edx = calculated free pages
3044 ; ecx = 0
3045 00004154 A1[A8760100] mov eax, [free_pages]
3046 00004159 39D0 cmp eax, edx ; calculated free mem value
3047 ; and initial free mem value are same or not?
3048 0000415B 751D jne short pmim ; print mem info with '?' if not
3049 0000415D 52 push edx ; free memory in pages
3050 ;mov eax, edx
3051 0000415E C1E00C shl eax, 12 ; convert page count
3052 ; to byte count
3053 00004161 B10A mov cl, 10
3054 00004163 BE[A5380100] mov esi, free_mem_b_str
3055 00004168 E8AD000000 call bintdstr
3056 0000416D 58 pop eax
3057 0000416E B107 mov cl, 7
3058 00004170 BE[C9380100] mov esi, free_mem_p_str
3059 00004175 E8A0000000 call bintdstr
3060 pmim:
3061 0000417A BE[4F380100] mov esi, msg_memory_info
3062 ;
3063 0000417F B407 mov ah, 07h ; Black background,
3064 ; light gray forecolor
3065 print_kmsg: ; 29/01/2016
3066 00004181 8825[CF760100] mov [ccolor], ah
3067 pkmsg_loop:
3068 00004187 AC lodsb
3069 00004188 08C0 or al, al
3070 0000418A 7410 jz short pkmsg_ok
3071 0000418C 56 push esi
3072 ; 13/05/2016
3073 0000418D 0FB61D[CF760100] movzx ebx, byte [ccolor]
3074 ; Video page 0 (bh=0)
3075 00004194 E842E1FFFF call _write_tty
3076 00004199 5E pop esi
3077 0000419A EBEB jmp short pkmsg_loop
3078 pkmsg_ok:
3079 0000419C C3 retn
3080
3081 ; 19/12/2020
3082 ; temporary
3083 ; write default liner frame buffer address
3084 ;
3085 default_lfb_info:
3086 0000419D 66A1[1A0F0000] mov ax, [def_LFB_addr] ; high word
3087 ; 24/11/2023 - temporary
3088 000041A3 6609C0 or ax, ax
3089 000041A6 74F4 jz short pkmsg_ok
3090 000041A8 E829000000 call wordtohex
3091 000041AD A3[21390100] mov dword [lfb_addr_str], eax
3092 000041B2 BE[0A390100] mov esi, msg_lfb_addr
3093 000041B7 B40F mov ah, 0Fh ; Black background,
3094 ; white forecolor
3095 000041B9 EBC6 jmp short print_kmsg
3096
3097 ; Convert binary number to hexadecimal string

```

```

3098 ; 10/05/2015
3099 ; dsectpm.s (28/02/2015)
3100 ; Retro UNIX 386 v1 - Kernel v0.2.0.6
3101 ; 01/12/2014
3102 ; 25/11/2014
3103 ;
3104 ;
3105 ; INPUT ->
3106 ; AL = byte (binary number)
3107 ; OUTPUT ->
3108 ; AX = hexadecimal string
3109 ;
3110 000041BB 53      push    ebx
3111 000041BC 31DB    xor     ebx, ebx
3112 000041BE 88C3    mov     bl, al
3113 000041C0 C0EB04    shr     bl, 4
3114 000041C3 8A9B[09420000]  mov     bl, [ebx+hexchrs]
3115 000041C9 86C3    xchg    bl, al
3116 000041CB 80E30F    and     bl, 0Fh
3117 000041CE 8AA3[09420000]  mov     ah, [ebx+hexchrs]
3118 000041D4 5B      pop     ebx
3119 000041D5 C3      retn
3120 ;
3121 ;
3122 ; INPUT ->
3123 ; AX = word (binary number)
3124 ; OUTPUT ->
3125 ; EAX = hexadecimal string
3126 ;
3127 000041D6 53      push    ebx
3128 000041D7 31DB    xor     ebx, ebx
3129 000041D9 86C4    xchg    ah, al
3130 000041DB 6650    push    ax ; * save ax
3131 000041DD 88E3    mov     bl, ah
3132 000041DF C0EB04    shr     bl, 4
3133 000041E2 8A83[09420000]  mov     al, [ebx+hexchrs]
3134 000041E8 88E3    mov     bl, ah
3135 000041EA 80E30F    and     bl, 0Fh
3136 000041ED 8AA3[09420000]  mov     ah, [ebx+hexchrs]
3137 000041F3 C1E010    shl     eax, 16 ; ax -> hw of eax
3138 000041F6 6658    pop     ax ; * restore ax
3139 000041F8 5B      pop     ebx
3140 000041F9 EBC0    jmp     short bytetohe
3141 ;mov     bl, al
3142 ;shr     bl, 4
3143 ;mov     bl, [ebx+hexchrs]
3144 ;xchg    bl, al
3145 ;and     bl, 0Fh
3146 ;mov     ah, [ebx+hexchrs]
3147 ;pop     ebx
3148 ;retn
3149 ;
3150 ;
3151 ; INPUT ->
3152 ; EAX = dword (binary number)
3153 ; OUTPUT ->
3154 ; EDX:EAX = hexadecimal string
3155 ;
3156 000041FB 50      push    eax
3157 000041FC C1E810    shr     eax, 16
3158 000041FF E8D2FFFFFF    call   wordtohex
3159 00004204 89C2    mov     edx, eax
3160 00004206 58      pop     eax
3161 ;call   wordtohex
3162 ;retn
3163 ; 18/04/2021
3164 00004207 EBCD    jmp     short wordtohex
3165 ;
3166 ; 10/05/2015
3167 hex_digits:
3168 hexchrs:
3169 00004209 303132333435363738- db '0123456789ABCDEF'
3169 00004212 39414243444546
3170 ;
3171 00004219 00      ; 19/01/2021 - VESA EDID ready flag (4Fh)
3172 edid: db 0
3173 ;
3174 ; Convert binary number to decimal/numeric string
3175 ; 06/11/2014
3176 ; Temporary Code
3177 ;
3178 ;
3179 ; EAX = binary number
3180 ; ESI = decimal/numeric string address
3181 ; EBX = divisor (10)
3182 ; ECX = string length (<=10)
3183 0000421A 01CE    add     esi, ecx
3184 ;
3185 0000421C 4E      dec     esi
3186 0000421D 31D2    xor     edx, edx
3187 0000421F F7F3    div     ebx
3188 00004221 80C230    add     dl, 30h
3189 00004224 8816    mov     [esi], dl
3190 00004226 FEC9    dec     cl
3191 00004228 740C    jz      short btdstr2 ; 08/09/2016
3192 0000422A 09C0    or      eax, eax
3193 0000422C 75EE    jnz     short btdstr0
3194 ;
3195 0000422E 4E      dec     esi
3196 0000422F C60620    mov     byte [esi], 20h ; blank space
3197 00004232 FEC9    dec     cl
3198 00004234 75F8    jnz     short btdstr1
3199 ;
3200 00004236 C3      retn
3201 ;
3202 ; Calculate free memory pages on M.A.T.
3203 ; 06/11/2014
3204 ; Temporary Code
3205 ;
3206 ;
3207 ;
3208 00004237 31D2    xor     edx, edx
3209 ;xor     ecx, ecx
3210 00004239 668B0D[B8760100]  mov     cx, [mat_size] ; in pages
3211 00004240 C1E10A    shl     ecx, 10 ; 1024 dwords per page
3212 00004243 BE00001000  mov     esi, MEM_ALLOC_TBL
3213 ;
3214 00004248 AD      lodsd
3215 00004249 51      push    ecx
3216 0000424A B920000000  mov     ecx, 32
3217 ;
3218 0000424F D1E8    shr     eax, 1
3219 00004251 7301    jnc     short cfm2
3220 00004253 42      inc     edx

```

```

3221      cfm2:
3222      loop   cfm1
3223      pop    ecx
3224      loop   cfm0
3225      retn
3226
3227      %include 'diskio.s' ; 07/03/2015
3228      ; *****
3229      ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.8 - diskio.s
3230      ; -----
3231      ; Last Update: 23/06/2024 (Previous: 02/12/2023 - Kernel v2.0.7)
3232      ; -----
3233      ; Beginning: 24/01/2016
3234      ; -----
3235      ; Assembler: NASM version 2.15 (trdos386.s)
3236      ; -----
3237      ; Turkish Rational DOS
3238      ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
3239      ; -----
3240      ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
3241      ; diskio.inc (22/08/2015)
3242      ; -----
3243      ; Derived from 'IBM PC-XT-286' BIOS source code (1986)
3244      ; *****
3245      ; Ref: Retro UNIX 386 v1.1 (Kernel v0.2.1.5) 'diskio' modification: 12/07/2022
3246      ; -----
3247      ; Retro UNIX 386 v1 Kernel - DISKIO.INC
3248      ; Last Modification: 22/08/2015
3249      ; (Initialized Disk Parameters Data is in 'DISKDATA.INC')
3250      ; (Uninitialized Disk Parameters Data is in 'DISKBSS.INC')
3251      ; -----
3252      ; DISK I/O SYSTEM - Erdogan Tan (Retro UNIX 386 v1 project)
3253      ; -----
3254      ; ////////// DISK I/O SYSTEM //////////
3255      ; -----
3256      ; 11/04/2021
3257      ; 06/02/2015
3258      ; diskette_io:
3259      ;   ;clc ; 20/07/2020
3260      ;   ;pushfd
3261      ;   ;push cs
3262      ;   ;call DISKETTE_IO_1
3263      ;   ;retn
3264      ; -----
3265      ; ;;;;; DISKETTE I/O ;;;;;; 06/02/2015 ;;;
3266      ; //////////
3267      ; -----
3268      ; DISKETTE I/O - Erdogan Tan (Retro UNIX 386 v1 project)
3269      ; 20/02/2015
3270      ; 06/02/2015 (unix386.s)
3271      ; 16/12/2014 - 02/01/2015 (dsectrm2.s)
3272      ; -----
3273      ; Code (DELAY) modifications - AWARD BIOS 1999 (ADISK.EQU, COMMON.MAC)
3274      ; -----
3275      ; ADISK.EQU
3276      ; -----
3277      ; ----- Wait control constants
3278      ; -----
3279      ; amount of time to wait while RESET is active.
3280      ; -----
3281      ; WAITCPU_RESET_ON EQU 21 ;Reset on must last at least 14us
3282      ; ;at 250 KBS xfer rate.
3283      ; ;see INTEL MCS, 1985, pg. 5-456
3284      ; -----
3285      ; WAITCPU_FOR_STATUS EQU 100 ;allow 30 microseconds for
3286      ; ;status register to become valid
3287      ; ;before re-reading.
3288      ; -----
3289      ; After sending a byte to NEC, status register may remain
3290      ; ;incorrectly set for 24 us.
3291      ; -----
3292      ; WAITCPU_RQM_LOW EQU 24 ;number of loops to check for
3293      ; ;RQM low.
3294      ; -----
3295      ; COMMON.MAC
3296      ; -----
3297      ; Timing macros
3298      ; -----
3299      ; -----
3300      ; %macro SIODELAY 0 ; SHORT IODELAY
3301      ; jmp short $+2
3302      ; %endmacro
3303      ; -----
3304      ; %macro IODELAY 0 ; NORMAL IODELAY
3305      ; jmp short $+2
3306      ; jmp short $+2
3307      ; %endmacro
3308      ; -----
3309      ; %macro NEWIODELAY 0
3310      ; out 0EBh, al
3311      ; %endmacro
3312      ; -----
3313      ; (According to) AWARD BIOS 1999 - ATORGS.ASM (dw -> equ, db -> equ)
3314      ; ;;; WAIT_FOR_MEM
3315      ; WAIT_FDU_INT_LO equ 017798 ; 2.5 secs in 30 micro units.
3316      ; WAIT_FDU_INT_HI equ 1
3317      ; WAIT_FDU_INT_LH equ 83334 ; 27/02/2015 (2.5 seconds waiting)
3318      ; ;;; WAIT_FOR_PORT
3319      ; WAIT_FDU_SEND_LO equ 16667 ; .5 secons in 30 us units.
3320      ; WAIT_FDU_SEND_HI equ 0
3321      ; WAIT_FDU_SEND_LH equ 16667 ; 27/02/2015
3322      ; Time to wait while waiting for each byte of NEC results = .5
3323      ; seconds. .5 seconds = 500,000 micros. 500,000/30 = 16,667.
3324      ; WAIT_FDU_RESULTS_LO equ 16667 ; .5 seconds in 30 micro units.
3325      ; WAIT_FDU_RESULTS_HI equ 0
3326      ; WAIT_FDU_RESULTS_LH equ 16667 ; 27/02/2015
3327      ; ;;; WAIT_REFRESH
3328      ; amount of time to wait for head settle, per unit in parameter
3329      ; ;table = 1 ms.
3330      ; WAIT_FDU_HEAD_SETTLE equ 33 ; 1 ms in 30 micro units.
3331      ; -----
3332      ; ////////// DISKETTE I/O //////////
3333      ; -----
3334      ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - POSTEQU.INC)
3335      ; -----
3336      ; -----
3337      ; EQUATES USED BY POST AND BIOS :
3338      ; -----
3339      ; -----
3340      ; ----- 8042 KEYBOARD INTERFACE AND DIAGNOSTIC CONTROL REGISTERS -----
3341      ; PORT_A EQU 060H ; 8042 KEYBOARD SCAN CODE/CONTROL PORT
3342      ; PORT_B EQU 061H ; PORT B READ/WRITE DIAGNOSTIC REGISTER
3343      ; REFRESH_BIT EQU 00010000B ; REFRESH TEST BIT

```

```

118 <1>
119 <1>
120 <1> ;----- CMOS EQUATES FOR THIS SYSTEM : -----
121 <1>
122 <1> ;CMOS_PORT EQU 070H ; I/O ADDRESS OF CMOS ADDRESS PORT
123 <1> ;CMOS_DATA EQU 071H ; I/O ADDRESS OF CMOS DATA PORT
124 <1> ;NMI EQU 10000000B ; DISABLE NMI INTERRUPTS MASK -
125 <1> ; HIGH BIT OF CMOS LOCATION ADDRESS
126 <1>
127 <1> ;----- CMOS TABLE LOCATION ADDRESS'S ## -----
128 <1> CMOS_DISKETTE EQU 010H ; DISKETTE DRIVE TYPE BYTE ;
129 <1> ; EQU 011H ; - RESERVED ;C
130 <1> CMOS_DISK EQU 012H ; FIXED DISK TYPE BYTE ;H
131 <1> ; EQU 013H ; - RESERVED ;E
132 <1> CMOS_EQUIP EQU 014H ; EQUIPMENT WORD LOW BYTE ;C
133 <1>
134 <1> ;----- DISKETTE EQUATES -----
135 <1> INT_FLAG EQU 10000000B ; INTERRUPT OCCURRENCE FLAG
136 <1> DSK_CHG EQU 10000000B ; DISKETTE CHANGE FLAG MASK BIT
137 <1> DETERMINED EQU 00010000B ; SET STATE DETERMINED IN STATE BITS
138 <1> HOME EQU 00010000B ; TRACK 0 MASK
139 <1> SENSE_DRV_ST EQU 00000100B ; SENSE DRIVE STATUS COMMAND
140 <1> TRK_SLAP EQU 030H ; CRASH STOP (48 TPI DRIVES)
141 <1> QUIET_SEEK EQU 00AH ; SEEK TO TRACK 10
142 <1> ;MAX_DRV EQU 2 ; MAX NUMBER OF DRIVES
143 <1> HD12_SETTLE EQU 15 ; 1.2 M HEAD SETTLE TIME
144 <1> HD320_SETTLE EQU 20 ; 320 K HEAD SETTLE TIME
145 <1> MOTOR_WAIT EQU 37 ; 2 SECONDS OF COUNTS FOR MOTOR TURN OFF
146 <1>
147 <1> ;----- DISKETTE ERRORS -----
148 <1> ;TIME_OUT EQU 080H ; ATTACHMENT FAILED TO RESPOND
149 <1> ;BAD_SEEK EQU 040H ; SEEK OPERATION FAILED
150 <1> ;BAD_NEC EQU 020H ; DISKETTE CONTROLLER HAS FAILED
151 <1> ;BAD_CRC EQU 010H ; BAD CRC ON DISKETTE READ
152 <1> ;MED_NOT_FND EQU 00CH ; MEDIA TYPE NOT FOUND
153 <1> ;DMA_BOUNDARY EQU 009H ; ATTEMPT TO DMA ACROSS 64K BOUNDARY
154 <1> ;BAD_DMA EQU 008H ; DMA OVERRUN ON OPERATION
155 <1> ;MEDIA_CHANGE EQU 006H ; MEDIA REMOVED ON DUAL ATTACH CARD
156 <1> ;RECORD_NOT_FND EQU 004H ; REQUESTED SECTOR NOT FOUND
157 <1> ;WRITE_PROTECT EQU 003H ; WRITE ATTEMPTED ON WRITE PROTECT DISK
158 <1> ;BAD_ADDR_MARK EQU 002H ; ADDRESS MARK NOT FOUND
159 <1> ;BAD_CMD EQU 001H ; BAD COMMAND PASSED TO DISKETTE I/O
160 <1>
161 <1> ;----- DISK CHANGE LINE EQUATES -----
162 <1> NOCHGLN EQU 001H ; NO DISK CHANGE LINE AVAILABLE
163 <1> CHGLN EQU 002H ; DISK CHANGE LINE AVAILABLE
164 <1>
165 <1> ;----- MEDIA/DRIVE STATE INDICATORS -----
166 <1> TRK_CAPA EQU 00000001B ; 80 TRACK CAPABILITY
167 <1> FMT_CAPA EQU 00000010B ; MULTIPLE FORMAT CAPABILITY (1.2M)
168 <1> DRV_DET EQU 00000100B ; DRIVE DETERMINED
169 <1> MED_DET EQU 00010000B ; MEDIA DETERMINED BIT
170 <1> DBL_STEP EQU 00100000B ; DOUBLE STEP BIT
171 <1> RATE_MSK EQU 11000000B ; MASK FOR CLEARING ALL BUT RATE
172 <1> RATE_500 EQU 00000000B ; 500 KBS DATA RATE
173 <1> RATE_300 EQU 01000000B ; 300 KBS DATA RATE
174 <1> RATE_250 EQU 10000000B ; 250 KBS DATA RATE
175 <1> STRT_MSK EQU 00001100B ; OPERATION START RATE MASK
176 <1> SEND_MSK EQU 11000000B ; MASK FOR SEND RATE BITS
177 <1>
178 <1> ;----- MEDIA/DRIVE STATE INDICATORS COMPATIBILITY -----
179 <1> M3D3U EQU 00000000B ; 360 MEDIA/DRIVE NOT ESTABLISHED
180 <1> M3D1U EQU 00000001B ; 360 MEDIA,1.2DRIVE NOT ESTABLISHED
181 <1> M1D1U EQU 00000010B ; 1.2 MEDIA/DRIVE NOT ESTABLISHED
182 <1> MED_UNK EQU 00000111B ; NONE OF THE ABOVE
183 <1>
184 <1> ;----- INTERRUPT EQUATES -----
185 <1> ;EOI EQU 020H ; END OF INTERRUPT COMMAND TO 8259
186 <1> ;INTA00 EQU 020H ; 8259 PORT
187 <1> ;INTA01 EQU 021H ; 8259 PORT
188 <1> ;INTB00 EQU 0A0H ; 2ND 8259
189 <1> ;INTB01 EQU 0A1H ;
190 <1>
191 <1> ;-----
192 <1> DMA08 EQU 008H ; DMA STATUS REGISTER PORT ADDRESS
193 <1> DMA EQU 000H ; DMA CH.0 ADDRESS REGISTER PORT ADDRESS
194 <1> DMA18 EQU 0D0H ; 2ND DMA STATUS PORT ADDRESS
195 <1> DMA1 EQU 0C0H ; 2ND DMA CH.0 ADDRESS REGISTER ADDRESS
196 <1>
197 <1> ;TIMER EQU 040H ; 8254 TIMER - BASE ADDRESS
198 <1>
199 <1> ;-----
200 <1> DMA_PAGE EQU 081H ; START OF DMA PAGE REGISTERS
201 <1>
202 <1> ; 06/02/2015 (unix386.s, protected mode modifications)
203 <1> ; (unix386.s <-- dsectrm2.s)
204 <1> ; 11/12/2014 (copy from IBM PC-XT Model 286 BIOS - DSEG.INC)
205 <1>
206 <1> ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
207 <1> ; 10/12/2014
208 <1>
209 <1> ;int40h:
210 <1> ; pushf
211 <1> ; push cs
212 <1> ; cli
213 <1> ; call DISKETTE_IO_1
214 <1> ; retn
215 <1>
216 <1> ; DSJETTE ----- 04/21/86 DISKETTE BIOS
217 <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
218 <1> ;
219 <1> ;
220 <1> ;-- INT13H -----
221 <1> ; DISKETTE I/O
222 <1> ; THIS INTERFACE PROVIDES ACCESS TO THE 5 1/4 INCH 360 KB,
223 <1> ; 1.2 MB, 720 KB AND 1.44 MB DISKETTE DRIVES.
224 <1> ; INPUT
225 <1> ; (AH) = 00H RESET DISKETTE SYSTEM
226 <1> ; HARD RESET TO NEC, PREPARE COMMAND, RECALIBRATE REQUIRED
227 <1> ; ON ALL DRIVES
228 <1> ;-----
229 <1> ; (AH)= 01H READ THE STATUS OF THE SYSTEM INTO (AH)
230 <1> ; @DISKETTE_STATUS FROM LAST OPERATION IS USED
231 <1> ;-----
232 <1> ; REGISTERS FOR READ/WRITE/VERIFY/FORMAT
233 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
234 <1> ; (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
235 <1> ; (CH) - TRACK NUMBER (NOT VALUE CHECKED)
236 <1> ; MEDIA DRIVE TRACK NUMBER
237 <1> ; 320/360 320/360 0-39
238 <1> ; 320/360 1.2M 0-39
239 <1> ; 1.2M 1.2M 0-79
240 <1> ; 720K 720K 0-79
241 <1> ; 1.44M 1.44M 0-79

```

```

242 <1> ; (CL) - SECTOR NUMBER (NOT VALUE CHECKED, NOT USED FOR FORMAT)
243 <1> ; MEDIA DRIVE SECTOR NUMBER
244 <1> ; 320/360 320/360 1-8/9
245 <1> ; 320/360 1.2M 1-8/9
246 <1> ; 1.2M 1.2M 1-15
247 <1> ; 720K 720K 1-9
248 <1> ; 1.44M 1.44M 1-18
249 <1> ; (AL) NUMBER OF SECTORS (NOT VALUE CHECKED)
250 <1> ; MEDIA DRIVE MAX NUMBER OF SECTORS
251 <1> ; 320/360 320/360 8/9
252 <1> ; 320/360 1.2M 8/9
253 <1> ; 1.2M 1.2M 15
254 <1> ; 720K 720K 9
255 <1> ; 1.44M 1.44M 18
256 <1> ;
257 <1> ; (ES:BX) - ADDRESS OF BUFFER (NOT REQUIRED FOR VERIFY)
258 <1> ;
259 <1> ; -----
260 <1> ; (AH)= 02H READ THE DESIRED SECTORS INTO MEMORY
261 <1> ; -----
262 <1> ; (AH)= 03H WRITE THE DESIRED SECTORS FROM MEMORY
263 <1> ; -----
264 <1> ; (AH)= 04H VERIFY THE DESIRED SECTORS
265 <1> ; -----
266 <1> ; (AH)= 05H FORMAT THE DESIRED TRACK
267 <1> ; (ES,BX) MUST POINT TO THE COLLECTION OF DESIRED ADDRESS FIELDS
268 <1> ; FOR THE TRACK. EACH FIELD IS COMPOSED OF 4 BYTES, (C,H,R,N),
269 <1> ; WHERE C = TRACK NUMBER, H=HEAD NUMBER, R = SECTOR NUMBER,
270 <1> ; N= NUMBER OF BYTES PER SECTOR (00=128,01=256,02=512,03=1024),
271 <1> ; THERE MUST BE ONE ENTRY FOR EVERY SECTOR ON THE TRACK.
272 <1> ; THIS INFORMATION IS USED TO FIND THE REQUESTED SECTOR DURING
273 <1> ; READ/WRITE ACCESS.
274 <1> ; PRIOR TO FORMATTING A DISKETTE, IF THERE EXISTS MORE THAN
275 <1> ; ONE SUPPORTED MEDIA FORMAT TYPE WITHIN THE DRIVE IN QUESTION,
276 <1> ; THEN "SET DASD TYPE" (INT 13H, AH = 17H) OR 'SET MEDIA TYPE'
277 <1> ; (INT 13H, AH = 18H) MUST BE CALLED TO SET THE DISKETTE TYPE
278 <1> ; THAT IS TO BE FORMATTED. IF "SET DASD TYPE" OR "SET MEDIA TYPE"
279 <1> ; IS NOT CALLED, THE FORMAT ROUTINE WILL ASSUME THE
280 <1> ; MEDIA FORMAT TO BE THE MAXIMUM CAPACITY OF THE DRIVE.
281 <1> ;
282 <1> ; THESE PARAMETERS OF DISK BASE MUST BE CHANGED IN ORDER TO
283 <1> ; FORMAT THE FOLLOWING MEDIAS:
284 <1> ; -----
285 <1> ; : MEDIA : DRIVE : PARM 1 : PARM 2 :
286 <1> ; -----
287 <1> ; : 320K : 320K/360K/1.2M : 50H : 8 :
288 <1> ; : 360K : 320K/360K/1.2M : 50H : 9 :
289 <1> ; : 1.2M : 1.2M : 54H : 15 :
290 <1> ; : 720K : 720K/1.44M : 50H : 9 :
291 <1> ; : 1.44M : 1.44M : 6CH : 18 :
292 <1> ; -----
293 <1> ; NOTES: - PARM 1 = GAP LENGTH FOR FORMAT
294 <1> ; - PARM 2 = EOT (LAST SECTOR ON TRACK)
295 <1> ; - DISK BASE IS POINTED BY DISK POINTER LOCATED
296 <1> ; AT ABSOLUTE ADDRESS 0:78.
297 <1> ; - WHEN FORMAT OPERATIONS ARE COMPLETE, THE PARAMETERS
298 <1> ; SHOULD BE RESTORED TO THEIR RESPECTIVE INITIAL VALUES.
299 <1> ; -----
300 <1> ; (AH) = 08H READ DRIVE PARAMETERS
301 <1> ; REGISTERS
302 <1> ; INPUT
303 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
304 <1> ; ** 27/05/2016 - TRDOS 386 (TRDOS v2.0) **
305 <1> ; ** EBX = Buffer address for floppy disk parameters table **
306 <1> ;
307 <1> ; OUTPUT
308 <1> ; (ES:DI) POINTS TO DRIVE PARAMETER TABLE
309 <1> ; *** TRDOS 386 note: floppy disk parameter table (16 bytes)
310 <1> ; will be returned to user in EBX, buffer address *** 27/05/2016 ***
311 <1> ;
312 <1> ; (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM NUMBER OF TRACKS
313 <1> ; (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
314 <1> ; BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
315 <1> ; (DH) - MAXIMUM HEAD NUMBER
316 <1> ; (DL) - NUMBER OF DISKETTE DRIVES INSTALLED
317 <1> ; (BH) - 0
318 <1> ; (BL) - BITS 7 THRU 4 - 0
319 <1> ; BITS 3 THRU 0 - VALID DRIVE TYPE VALUE IN CMOS
320 <1> ; (AX) - 0
321 <1> ; UNDER THE FOLLOWING CIRCUMSTANCES:
322 <1> ; (1) THE DRIVE NUMBER IS INVALID,
323 <1> ; (2) THE DRIVE TYPE IS UNKNOWN AND CMOS IS NOT PRESENT,
324 <1> ; (3) THE DRIVE TYPE IS UNKNOWN AND CMOS IS BAD,
325 <1> ; (4) OR THE DRIVE TYPE IS UNKNOWN AND THE CMOS DRIVE TYPE IS INVALID
326 <1> ; THEN ES,AX,BX,CX,DH,DI=0 ; DL=NUMBER OF DRIVES.
327 <1> ; IF NO DRIVES ARE PRESENT THEN: ES,AX,BX,CX,DX,DI=0.
328 <1> ; @DISKETTE_STATUS = 0 AND CY IS RESET.
329 <1> ; -----
330 <1> ; (AH)= 15H READ DASD TYPE
331 <1> ; OUTPUT REGISTERS
332 <1> ; (AH) - ON RETURN IF CARRY FLAG NOT SET, OTHERWISE ERROR
333 <1> ; 00 - DRIVE NOT PRESENT
334 <1> ; 01 - DISKETTE, NO CHANGE LINE AVAILABLE
335 <1> ; 02 - DISKETTE, CHANGE LINE AVAILABLE
336 <1> ; 03 - RESERVED (FIXED DISK)
337 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
338 <1> ; -----
339 <1> ; (AH)= 16H DISK CHANGE LINE STATUS
340 <1> ; OUTPUT REGISTERS
341 <1> ; (AH) - 00 - DISK CHANGE LINE NOT ACTIVE
342 <1> ; 06 - DISK CHANGE LINE ACTIVE & CARRY BIT ON
343 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED)
344 <1> ; -----
345 <1> ; (AH)= 17H SET DASD TYPE FOR FORMAT
346 <1> ; INPUT REGISTERS
347 <1> ; (AL) - 00 - NOT USED
348 <1> ; 01 - DISKETTE 320/360K IN 360K DRIVE
349 <1> ; 02 - DISKETTE 360K IN 1.2M DRIVE
350 <1> ; 03 - DISKETTE 1.2M IN 1.2M DRIVE
351 <1> ; 04 - DISKETTE 720K IN 720K DRIVE
352 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHECKED:
353 <1> ; (DO NOT USE WHEN DISKETTE ATTACH CARD USED)
354 <1> ; -----
355 <1> ; (AH)= 18H SET MEDIA TYPE FOR FORMAT
356 <1> ; INPUT REGISTERS
357 <1> ; (CH) - LOW ORDER 8 OF 10 BITS MAXIMUM TRACKS
358 <1> ; (CL) - BITS 7 & 6 - HIGH ORDER TWO BITS OF MAXIMUM TRACKS
359 <1> ; BITS 5 THRU 0 - MAXIMUM SECTORS PER TRACK
360 <1> ; (DL) - DRIVE NUMBER (0-1 ALLOWED, VALUE CHACKED)
361 <1> ; OUTPUT REGISTERS:
362 <1> ; (ES:DI) - POINTER TO DRIVE PARAMETERS TABLE FOR THIS MEDIA TYPE,
363 <1> ; UNCHANGED IF (AH) IS NON-ZERO
364 <1> ; (AH) - 00H, CY = 0, TRACK AND SECTORS/TRACK COMBINATION IS SUPPORTED
365 <1> ; - 01H, CY = 1, FUNCTION IS NOT AVAILABLE
366 <1> ; - 0CH, CY = 1, TRACK AND SECTORS/TRACK COMBINATION IS NOT SUPPORTED

```

```

366 <1> ; - 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)
367 <1> ;-----
368 <1> ; DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER
369 <1> ; THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE
370 <1> ; ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:
371 <1> ;     ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
372 <1> ;     IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK
373 <1> ;     CHANGE ERROR CODE
374 <1> ;     IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE
375 <1> ;     TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.
376 <1> ; IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.
377 <1> ;
378 <1> ; DATA VARIABLE -- @DISK_POINTER
379 <1> ; DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS
380 <1> ;-----
381 <1> ; OUTPUT FOR ALL FUNCTIONS
382 <1> ; AH = STATUS OF OPERATION
383 <1> ;     STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE_STATUS
384 <1> ;     VARIABLE IN THE DATA SEGMENT OF THIS MODULE
385 <1> ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD
386 <1> ;     TYPE AH=(15)).
387 <1> ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON)
388 <1> ; FOR READ/WRITE/VERIFY
389 <1> ;     DS,BX,DX,CX PRESERVED
390 <1> ; NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE
391 <1> ; ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION.
392 <1> ; ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT
393 <1> ; THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE
394 <1> ; PROBLEM IS NOT DUE TO MOTOR START-UP.
395 <1> ;-----
396 <1> ;
397 <1> ; DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)
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> ;
430 <1> ;
431 <1> ;
432 <1> ;
433 <1> ;
434 <1> ;
435 <1> ;
436 <1> ;
437 <1> ;
438 00000000 ??
439 00000001 ??
440 00000002 ??
441 00000003 ??
442 00000004 ??
443 00000005 ??
444 00000006 ??
445 00000007 ??
446 00000008 ??
447 00000009 ??
448 0000000A ??
449 0000000B ??
450 0000000C ??
451 <1> ;
452 <1> ;
453 <1> ;
454 <1> ;
455 <1> ;
456 <1> ;
457 <1> ;
458 <1> ;
459 <1> ;
460 <1> ;
461 <1> ;
462 <1> ;
463 <1> ;
464 <1> ;
465 <1> ;
466 <1> ;
467 <1> ;
468 <1> ;
469 <1> ;
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 <1> ;
484 <1> ;
485 <1> ;
486 <1> ;
487 <1> ;
488 <1> ;
489 <1> ;

```

- 80H, CY = 1, TIME OUT (DISKETTE NOT PRESENT)

DISK CHANGE STATUS IS ONLY CHECKED WHEN A MEDIA SPECIFIED IS OTHER THAN 360 KB DRIVE. IF THE DISK CHANGE LINE IS FOUND TO BE ACTIVE THE FOLLOWING ACTIONS TAKE PLACE:

- ATTEMPT TO RESET DISK CHANGE LINE TO INACTIVE STATE.
- IF ATTEMPT SUCCEEDS SET DASD TYPE FOR FORMAT AND RETURN DISK CHANGE ERROR CODE
- IF ATTEMPT FAILS RETURN TIMEOUT ERROR CODE AND SET DASD TYPE TO A PREDETERMINED STATE INDICATING MEDIA TYPE UNKNOWN.

IF THE DISK CHANGE LINE IN INACTIVE PERFORM SET DASD TYPE FOR FORMAT.

DATA VARIABLE -- @DISK\_POINTER  
 DOUBLE WORD POINTER TO THE CURRENT SET OF DISKETTE PARAMETERS

OUTPUT FOR ALL FUNCTIONS

AH = STATUS OF OPERATION

STATUS BITS ARE DEFINED IN THE EQUATES FOR @DISKETTE\_STATUS VARIABLE IN THE DATA SEGMENT OF THIS MODULE

CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN, EXCEPT FOR READ DASD TYPE AH=(15)).

CY = 1 FAILED OPERATION (AH HAS ERROR REASON)

FOR READ/WRITE/VERIFY

DS,BX,DX,CX PRESERVED

NOTE: IF AN ERROR IS REPORTED BY THE DISKETTE CODE, THE APPROPRIATE ACTION IS TO RESET THE DISKETTE, THEN RETRY THE OPERATION. ON READ ACCESSES, NO MOTOR START DELAY IS TAKEN, SO THAT THREE RETRIES ARE REQUIRED ON READS TO ENSURE THAT THE PROBLEM IS NOT DUE TO MOTOR START-UP.

DISKETTE STATE MACHINE - ABSOLUTE ADDRESS 40:90 (DRIVE A) & 91 (DRIVE B)

7	6	5	4	3	2	1	0
				PRESENT STATE			
				000: 360K IN 360K DRIVE UNESTABLISHED			
				001: 360K IN 1.2M DRIVE UNESTABLISHED			
				010: 1.2M IN 1.2M DRIVE UNESTABLISHED			
				011: 360K IN 360K DRIVE ESTABLISHED			
				100: 360K IN 1.2M DRIVE ESTABLISHED			
				101: 1.2M IN 1.2M DRIVE ESTABLISHED			
				110: RESERVED			
				111: NONE OF THE ABOVE			
				-----> MEDIA/DRIVE ESTABLISHED			
				-----> DOUBLE STEPPING REQUIRED (360K IN 1.2M DRIVE)			
				-----> DATA TRANSFER RATE FOR THIS DRIVE:			
				00: 500 KBS			
				01: 300 KBS			
				10: 250 KBS			
				11: RESERVED			

STATE OPERATION STARTED - ABSOLUTE ADDRESS 40:92 (DRIVE A) & 93 (DRIVE B)

PRESENT CYLINDER NUMBER - ABSOLUTE ADDRESS 40:94 (DRIVE A) & 95 (DRIVE B)

```

struc MD
    .SPEC1      resb 1      ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
    .SPEC2      resb 1      ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
    .OFF_TIM     resb 1      ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
    .BYT_SEC     resb 1      ; 512 BYTES/SECTOR
    .SEC_TRK     resb 1      ; EOT (LAST SECTOR ON TRACK)
    .GAP         resb 1      ; GAP LENGTH
    .DTL         resb 1      ; DTL
    .GAP3        resb 1      ; GAP LENGTH FOR FORMAT
    .FIL_BYT     resb 1      ; FILL BYTE FOR FORMAT
    .HD_TIM      resb 1      ; HEAD SETTLE TIME (MILLISECONDS)
    .STR_TIM     resb 1      ; MOTOR START TIME (1/8 SECONDS)
    .MAX_TRK     resb 1      ; MAX. TRACK NUMBER
    .RATE        resb 1      ; DATA TRANSFER RATE
endstruc

BIT7OFF EQU 7FH
BIT7ON  EQU 80H

; 30/08/2020 - TRDOS 386 v2

;;int13h: ; 16/02/2015
;; 16/02/2015 - 21/02/2015
; 17/07/2022 - TRDOS 386 v2.0.5
int40h:
; 11/04/2021
diskette_io:
    cld ; 20/07/2020
    pushfd
    push cs
    call DISKETTE_IO_1
    retn

; 09/08/2022
; 06/08/2022
; 17/07/2022 - TRDOS 386 v2.0.5
; (jump from DISK_IO)
DISKETTE_IO_1:
    sti ; 17/07/2022 ; INTERRUPTS BACK ON

    push ebp ; USER REGISTER
    push edi ; USER REGISTER
    push edx ; HEAD #, DRIVE # OR USER REGISTER
    push ebx ; BUFFER OFFSET PARAMETER OR REGISTER
    push ecx ; TRACK #-SECTOR # OR USER REGISTER
    mov ebp, esp ; BP => PARAMETER LIST DEP. ON AH
    [BP] = SECTOR #
    [BP+1] = TRACK #
    [BP+2] = BUFFER OFFSET
    FOR RETURN OF DRIVE PARAMETERS:
    CL/[BP] = BITS 7&6 HI BITS OF MAX CYL
    BITS 0-5 MAX SECTORS/TRACK
    CH/[BP+1] = LOW 8 BITS OF MAX CYL.

```

```

490      <1>                                ; BL/[BP+2] = BITS 7-4 = 0
491      <1>                                ;                                ; BITS 3-0 = VALID CMOS TYPE
492      <1>                                ; BH/[BP+3] = 0
493      <1>                                ; DL/[BP+4] = # DRIVES INSTALLED
494      <1>                                ; DH/[BP+5] = MAX HEAD #
495      <1>                                ; DI/[BP+6] = OFFSET TO DISK BASE
496      <1>                                ;
497      <1>                                ;
498      <1>                                ;
499      <1>                                ;
500      <1>                                ;
501      <1>                                ;
502      <1>                                ;
503      <1>                                ;
504      <1>                                ;
505      <1>                                ;
506      <1>                                ;
507      <1>                                ;
508      <1>                                ;
509      <1>                                ;
510      <1>                                ;
511      <1>                                ;
512      <1>                                ;
513      <1>                                ;
514      <1>                                ;
515      <1>                                ;
516      <1>                                ;
517      <1>                                ;
518      <1>                                ;
519      <1>                                ;
520      <1>                                ;
521      <1>                                ;
522      <1>                                ;
523      0000425A 55      <1>                                ;
524      0000425B 89DD      <1>                                ;
525      <1>                                ;
526      <1>                                ;
527      0000425D 80FC19      <1>                                ;
528      <1>                                ;
529      <1>                                ;
530      <1>                                ;
531      00004260 730F      <1>                                ;
532      <1>                                ;
533      00004262 80FC01      <1>                                ;
534      00004265 760C      <1>                                ;
535      00004267 80FC08      <1>                                ;
536      0000426A 7407      <1>                                ;
537      0000426C 80FA01      <1>                                ;
538      0000426F 7602      <1>                                ;
539      <1>                                ;
540      00004271 B414      <1>                                ;
541      <1>                                ;
542      <1>                                ;
543      <1>                                ;
544      <1>                                ;
545      <1>                                ;
546      <1>                                ;
547      <1>                                ;
548      <1>                                ;
549      <1>                                ;
550      <1>                                ;
551      <1>                                ;
552      <1>                                ;
553      <1>                                ;
554      00004273 29DB      <1>                                ;
555      00004275 88E3      <1>                                ;
556      00004277 C0E302      <1>                                ;
557      0000427A 81C3[99420000] <1>                                ;
558      <1>                                ;
559      00004280 88F4      <1>                                ;
560      <1>                                ;
561      <1>                                ;
562      <1>                                ;
563      <1>                                ;
564      00004282 0FB7F0      <1>                                ;
565      00004285 0FB6FA      <1>                                ;
566      <1>                                ;
567      <1>                                ;
568      <1>                                ;
569      <1>                                ;
570      <1>                                ;
571      <1>                                ;
572      <1>                                ;
573      <1>                                ;
574      <1>                                ;
575      <1>                                ;
576      00004288 8A25[28770100] <1>                                ;
577      0000428E C605[28770100]00 <1>                                ;
578      <1>                                ;
579      <1>                                ;
580      <1>                                ;
581      <1>                                ;
582      <1>                                ;
583      <1>                                ;
584      <1>                                ;
585      <1>                                ;
586      <1>                                ;
587      <1>                                ;
588      <1>                                ;
589      <1>                                ;
590      <1>                                ;
591      <1>                                ;
592      <1>                                ;
593      <1>                                ;
594      <1>                                ;
595      <1>                                ;
596      <1>                                ;
597      <1>                                ;
598      <1>                                ;
599      <1>                                ;
600      <1>                                ;
601      <1>                                ;
602      <1>                                ;
603      <1>                                ;
604      <1>                                ;
605      00004295 FF13      <1>                                ;
606      <1>                                ;
607      <1>                                ;
608      <1>                                ;
609      <1>                                ;
610      <1>                                ;
611      <1>                                ;
612      <1>                                ;
613      <1>                                ;

; BL/[BP+2] = BITS 7-4 = 0
;                                ; BITS 3-0 = VALID CMOS TYPE
; BH/[BP+3] = 0
; DL/[BP+4] = # DRIVES INSTALLED
; DH/[BP+5] = MAX HEAD #
; DI/[BP+6] = OFFSET TO DISK BASE
;
;
;
;
;
;
;
;
;
;
; 17/07/2022
; Registers are also on stack
; (with same contents)
; in following order:
;
;     ebx = esp+20
;     ecx = esp+16
;     edx = esp+12
;     esi = esp+8
;     edi = esp+4
;
; [esp] = caller's return address (from 'DISK_IO')
;
; cs = KCODE == KDATA
; ds = es = ss = KDATA
;
; 17/07/2022
; push    ebp
; mov     ebp, ebx
;
; cmp     ah, (FNC_TAE-FNC_TAB)/2      ; CHECK FOR > LARGEST FUNCTION
; cmp     ah, (FNC_TAE-FNC_TAB)/4      ; 18/02/2015
; jb      short OK_FUNC                ; FUNCTION OK
; mov     ah, 14h                      ; REPLACE WITH KNOWN INVALID FUNCTION
; 09/08/2022
; jnb     short INV_FUNC
;
; OK_FUNC:
; cmp     ah, 1                        ; RESET OR STATUS ?
; jbe     short OK_DRV                ; IF RESET OR STATUS DRIVE ALWAYS OK
; cmp     ah, 8                        ; READ DRIVE PARMS ?
; je      short OK_DRV                ; IF SO DRIVE CHECKED LATER
; cmp     dl, 1                        ; DRIVES 0 AND 1 OK
; jbe     short OK_DRV                ; IF 0 OR 1 THEN JUMP
;
; INV_FUNC:
; mov     ah, 14h                      ; REPLACE WITH KNOWN INVALID FUNCTION
;
; OK_DRV:
; 17/07/2022
; xor     ecx, ecx
; mov     esi, ecx ; 08/02/2015
; mov     edi, ecx ; 08/02/2015
; mov     cl, ah
; xor     ch, ch
; shl     cl, 1
; shl     cl, 2 ; 20/02/2015
; mov     ebx, FNC_TAB
; add     ebx, ecx
;
; CL = FUNCTION
; CX = FUNCTION
; FUNCTION TIMES 2
; FUNCTION TIMES 4 (for 32 bit offset)
; LOAD START OF FUNCTION TABLE
; ADD OFFSET INTO TABLE => ROUTINE
;
; 17/07/2022
; sub     ebx, ebx
; mov     bl, ah
; shl     bl, 2 ; * 4
; add     ebx, FNC_TAB
;
; BL = FUNCTION
; [EBX] = FUNCTION ADDRESS
;
; mov     ah, dh
; xor     dh, dh
; mov     si, ax
; mov     di, dx
;
; AX = HEAD #, # OF SECTORS OR DASD TYPE
; DX = DRIVE #
; SI = HEAD #, # OF SECTORS OR DASD TYPE
; DI = DRIVE #
;
; movzx   esi, ax
; movzx   edi, dl
;
; ESI = HEAD #, # OF SECTORS OR DASD TYPE
; EDI = DRIVE #
;
; CH = cylinder number (low 8 bit)
; CL = sector number (and high 2 bits of cylinder number)
;
; 06/08/2022
; 11/12/2014
; mov     [cfd], dl
;
; current floppy drive (for 'GET_PARM')
; 06/08/2022
; EDI = (current) DRIVE #
;
;
; mov     ah, [DSKETTE_STATUS] ; LOAD STATUS TO AH FOR STATUS FUNCTION
; mov     byte [DSKETTE_STATUS], 0 ; INITIALIZE FOR ALL OTHERS
;
;
; THROUGHOUT THE DISKETTE BIOS, THE FOLLOWING INFORMATION IS CONTAINED IN
; THE FOLLOWING MEMORY LOCATIONS AND REGISTERS. NOT ALL DISKETTE BIOS
; FUNCTIONS REQUIRE ALL OF THESE PARAMETERS.
;
; DI      : DRIVE #
; SI-HI   : HEAD #
; SI-LOW  : # OF SECTORS OR DASD TYPE FOR FORMAT
; ES      : BUFFER SEGMENT
; [BP]    : SECTOR #
; [BP+1]  : TRACK #
; [BP+2]  : BUFFER OFFSET
;
;
; ACROSS CALLS TO SUBROUTINES THE CARRY FLAG (CY=1), WHERE INDICATED IN
; SUBROUTINE PROLOGUES, REPRESENTS AN EXCEPTION RETURN (NORMALLY AN ERROR
; CONDITION). IN MOST CASES, WHEN CY = 1, @DSKETTE_STATUS CONTAINS THE
; SPECIFIC ERROR CODE.
;
; 17/07/2022
; EBX = pointer to function address
; EBP = buffer address
; EDI = drive number (0 or 1)
; ESI = head number (byte 1) and sector count or disk type (byte 0)
; CH = cylinder number (low 8 bit)
; CL = sector number (and high 2 bits of cylinder number)
;
; (AH) = @DSKETTE_STATUS
; CALL THE REQUESTED FUNCTION
;
; pop     esi
; pop     ds
; pop     es ; 06/02/2015
; pop     ecx
; pop     ebx
; pop     edx
; pop     edi

```

```

614      <1>      ;mov     ebp, esp
615      <1>      ;push    eax
616      <1>      ;pushfd
617      <1>      ;pop     eax
618      <1>      ;mov     [bp+6], ax
619      <1>      ;mov     [ebp+12], eax ; 18/02/2015, flags
620      <1>      ;pop     eax
621      <1>      ;pop     ebp
622      <1>      ;iretd
623      <1>
624      <1>      ; 17/07/2022
625      00004297 5D      <1>      pop     ebp
626      <1>
627      <1>      ; 17/07/2022
628      <1>      ; Stack order:
629      <1>      ;     ebx = esp+20
630      <1>      ;     ecx = esp+16
631      <1>      ;     edx = esp+12
632      <1>      ;     esi = esp+8
633      <1>      ;     edi = esp+4
634      <1>
635      <1>      ; [esp] = caller's return address (from 'DISK_IO')
636      <1>
637      <1>      ; CF = disk i/o status (1 = error)
638      <1>      ; AH = error code (if > 0 and cf = 1)
639      <1>
640      <1>      ; 17/07/2022
641      00004298 C3      <1>      retn    ; return to the caller of 'DISK_IO'
642      <1>
643      <1>
644      <1>      ;-----
645      <1>      ; DW --> dd (06/02/2015)
646      00004299 [FD420000] <1>      FNC_TAB    dd     DSK_RESET      ; AH = 00H; RESET
647      0000429D [68430000] <1>      dd         DSK_STATUS      ; AH = 01H; STATUS
648      000042A1 [78430000] <1>      dd         DSK_READ       ; AH = 02H; READ
649      000042A5 [85430000] <1>      dd         DSK_WRITE      ; AH = 03H; WRITE
650      000042A9 [99440000] <1>      dd         DSK_VERF       ; AH = 04H; VERIFY
651      000042AD [A9440000] <1>      dd         DSK_FORMAT      ; AH = 05H; FORMAT
652      000042B1 [27450000] <1>      dd         FNC_ERR        ; AH = 06H; INVALID
653      000042B5 [27450000] <1>      dd         FNC_ERR        ; AH = 07H; INVALID
654      000042B9 [33450000] <1>      dd         DSK_PARAMS      ; AH = 08H; READ DRIVE PARAMETERS
655      000042BD [27450000] <1>      dd         FNC_ERR        ; AH = 09H; INVALID
656      000042C1 [27450000] <1>      dd         FNC_ERR        ; AH = 0AH; INVALID
657      000042C5 [27450000] <1>      dd         FNC_ERR        ; AH = 0BH; INVALID
658      000042C9 [27450000] <1>      dd         FNC_ERR        ; AH = 0CH; INVALID
659      000042CD [27450000] <1>      dd         FNC_ERR        ; AH = 0DH; INVALID
660      000042D1 [27450000] <1>      dd         FNC_ERR        ; AH = 0EH; INVALID
661      000042D5 [27450000] <1>      dd         FNC_ERR        ; AH = 0FH; INVALID
662      000042D9 [27450000] <1>      dd         FNC_ERR        ; AH = 10H; INVALID
663      000042DD [27450000] <1>      dd         FNC_ERR        ; AH = 11H; INVALID
664      000042E1 [27450000] <1>      dd         FNC_ERR        ; AH = 12H; INVALID
665      000042E5 [27450000] <1>      dd         FNC_ERR        ; AH = 13H; INVALID
666      000042E9 [27450000] <1>      dd         FNC_ERR        ; AH = 14H; INVALID
667      000042ED [F0450000] <1>      dd         DSK_TYPE       ; AH = 15H; READ DASD TYPE
668      000042F1 [10460000] <1>      dd         DSK_CHANGE      ; AH = 16H; CHANGE STATUS
669      000042F5 [3F460000] <1>      dd         FORMAT_SET      ; AH = 17H; SET DASD TYPE
670      000042F9 [B4460000] <1>      dd         SET_MEDIA      ; AH = 18H; SET MEDIA TYPE
671      <1>      FNC_TAE EQU    $          ; END
672      <1>
673      <1>      ; 17/07/2022 - TRDOS 386 v2.0.5
674      <1>      ;-----
675      <1>      ; DISK_RESET      (AH = 00H)
676      <1>      ; RESET THE DISKETTE SYSTEM.
677      <1>      ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
678      <1>      ;-----
679      <1>      DSK_RESET:
680      <1>      ; 17/07/2022
681      000042FD 66BAF203 <1>      mov     dx, 03F2h      ; ADAPTER CONTROL PORT
682      00004301 FA      <1>      cli          ; NO INTERRUPTS
683      00004302 A0[26770100] <1>      mov     al, [MOTOR_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION
684      00004307 243F      <1>      and     al, 00111111b ; KEEP SELECTED AND MOTOR ON BITS
685      00004309 C0C004      <1>      rol     al, 4          ; MOTOR VALUE TO HIGH NIBBLE
686      <1>      <1>      ; DRIVE SELECT TO LOW NIBBLE
687      0000430C 0C08      <1>      or      al, 00001000b ; TURN ON INTERRUPT ENABLE
688      0000430E EE      <1>      out     dx, al      ; RESET THE ADAPTER
689      0000430F C605[25770100]00 <1>      mov     byte [SEEK_STATUS], 0 ; SET RECALIBRATE REQUIRED ON ALL DRIVES
690      <1>      ; jmp     $+2          ; WAIT FOR I/O
691      <1>      ; jmp     $+2          ; WAIT FOR I/O (TO INSURE MINIMUM
692      <1>      <1>      ; PULSE WIDTH)
693      <1>      ; 19/12/2014
694      <1>      NEWIODELAY
695      83 00004316 E6EB      <2>      out     0EBh, al
696      <1>
697      <1>      ; 17/12/2014
698      00004318 B915000000 <1>      ; AWARD BIOS 1999 - RESETDRIVES (ADISK.ASM)
699      <1>      mov     ecx, WAITCPU_RESET_ON ; cx = 21 -- Min. 14 micro seconds !?
700      <1>      wdw1: NEWIODELAY ; 27/02/2015
701      83 0000431D E6EB      <2>      out     0EBh, al
702      701 0000431F E2FC      <1>      loop    wdw1
703      <1>
704      <1>      ; or      al, 00000100b ; TURN OFF RESET BIT
705      00004321 0C04      <1>      out     dx, al      ; RESET THE ADAPTER
706      <1>      ; 16/12/2014
707      <1>      IODELAY
708      78 00004324 EB00      <2>      jmp     short $+2
709      79 00004326 EB00      <2>      jmp     short $+2
710      <1>
711      <1>      ; sti          ; ENABLE THE INTERRUPTS
712      00004328 E8DD0A0000 <1>      call    WAIT_INT      ; WAIT FOR THE INTERRUPT
713      0000432D 7230      <1>      jc      short DR_ERR ; IF ERROR, RETURN IT
714      <1>      ; mov     cx, 11000000b ; CL = EXPECTED @NEC_STATUS
715      <1>      ; 17/07/2022
716      <1>      ; xor     ch, ch
717      0000432F B1C0      <1>      mov     cl, 11000000b
718      <1>      NXT_DRV:
719      <1>      ; push    cx          ; SAVE FOR CALL
720      <1>      ; 11/04/2021
721      00004331 51      <1>      push    ecx
722      00004332 B8[5E430000] <1>      mov     eax, DR_POP_ERR ; LOAD NEC_OUTPUT ERROR ADDRESS
723      00004337 50      <1>      push    eax
724      00004338 B408      <1>      mov     ah, 08h ; SENSE INTERRUPT STATUS COMMAND
725      0000433A E8C3090000 <1>      call    NEC_OUTPUT
726      0000433F 58      <1>      pop     eax ; THROW AWAY ERROR RETURN
727      00004340 E8F40A0000 <1>      call    RESULTS ; READ IN THE RESULTS
728      <1>      ; pop     cx ; RESTORE AFTER CALL
729      <1>      ; 11/04/2021
730      00004345 59      <1>      pop     ecx
731      00004346 7217      <1>      jc      short DR_ERR ; ERROR RETURN
732      00004348 3A0D[29770100] <1>      cmp     cl, [NEC_STATUS] ; TEST FOR DRIVE READY TRANSITION
733      0000434E 750F      <1>      jnz     short DR_ERR ; EVERYTHING OK
734      00004350 FEC1      <1>      inc     cl ; NEXT EXPECTED @NEC_STATUS
735      00004352 80F9C3      <1>      cmp     cl, 11000011b ; ALL POSSIBLE DRIVES CLEARED
736      00004355 76DA      <1>      jbe     short NXT_DRV ; FALL THRU IF 11000100B OR >

```



```

734      ;
735 00004357 E800040000      <1>      call    SEND_SPEC          ; SEND SPECIFY COMMAND TO NEC
736      <1> RESBAC:
737      <1>      ; 06/08/2022
738 0000435C EB10          <1>      jmp     short SETUP_END_X
739      <1>      ;call    SETUP_END          ; VARIOUS CLEANUPS
740      <1>      ;;mov     bx, si          ; GET SAVED AL TO BL
741      <1>      ;; 17/07/2022
742      <1>      ;mov     ebx, esi
743      <1>      ;mov     al, bl          ; PUT BACK FOR RETURN
744      <1>      ;retn
745      <1>
746      <1> DR_POP_ERR:
747      <1>      ;pop     cx          ; CLEAR STACK
748      <1>      ; 11/04/2021
749 0000435E 59          <1>      pop     ecx
750      <1> DR_ERR:
751 0000435F 800D[28770100]20 <1>      or      byte [DSKETTE_STATUS], BAD_NEC ; SET ERROR CODE
752      <1>      ;jmp     short RESBAC          ; RETURN FROM RESET
753      <1>      ; 06/08/2022
754 00004366 EB06          <1>      jmp     short SETUP_END_X
755      <1>
756      <1> -----
757      <1> ; DISK_STATUS      (AH = 01H)
758      <1> ; DISKETTE STATUS.
759      <1> ;
760      <1> ; ON ENTRY: AH : STATUS OF PREVIOUS OPERATION
761      <1> ;
762      <1> ; ON EXIT: AH, @DSKETTE_STATUS, CY REFLECT STATUS OF PREVIOUS OPERATION.
763      <1> ; -----
764      <1> DSK_STATUS:
765 00004368 8825[28770100] <1>      mov     [DSKETTE_STATUS], ah      ; PUT BACK FOR SETUP END
766      <1>      SETUP_END_X:      ; 06/08/2022
767 0000436E E809010000 <1>      call    SETUP_END          ; VARIOUS CLEANUPS
768      <1>      ;mov     bx, si          ; GET SAVED AL TO BL
769      <1>      ;mov     al, bl          ; PUT BACK FOR RETURN
770      <1>      ; 06/08/2022
771 00004373 89F3          <1>      mov     ebx, esi
772 00004375 88D8          <1>      mov     al, bl
773 00004377 C3          <1>      retn
774      <1>
775      <1> -----
776      <1> ; DISK_READ (AH = 02H)
777      <1> ; DISKETTE READ.
778      <1> ;
779      <1> ; ON ENTRY: DI      : DRIVE #
780      <1> ; SI-HI      : HEAD #
781      <1> ; SI-LOW     : # OF SECTORS
782      <1> ; ES      : BUFFER SEGMENT
783      <1> ; [BP]      : SECTOR #
784      <1> ; [BP+1]    : TRACK #
785      <1> ; [BP+2]    : BUFFER OFFSET
786      <1> ;
787      <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
788      <1> ; -----
789      <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
790      <1> ;
791      <1> DSK_READ:
792      <1>      and     byte [MOTOR_STATUS], 01111111b ; INDICATE A READ OPERATION
793 00004378 8025[26770100]7F <1>      mov     ax, 0E646h          ; AX = NEC COMMAND, DMA COMMAND
794 0000437F 66B846E6 <1>      ;call    RD_WR_VF          ; COMMON READ/WRITE/VERIFY
795      <1>      ;retn
796      <1>      ; 06/08/2022
797      <1>      jmp     short RD_WR_VF
798 00004383 EB0B          <1>
799      <1>
800      <1> -----
801      <1> ; DISK_WRITE      (AH = 03H)
802      <1> ; DISKETTE WRITE.
803      <1> ;
804      <1> ; ON ENTRY: DI      : DRIVE #
805      <1> ; SI-HI      : HEAD #
806      <1> ; SI-LOW     : # OF SECTORS
807      <1> ; ES      : BUFFER SEGMENT
808      <1> ; [BP]      : SECTOR #
809      <1> ; [BP+1]    : TRACK #
810      <1> ; [BP+2]    : BUFFER OFFSET
811      <1> ;
812      <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
813      <1> ; -----
814      <1> ; 06/02/2015, ES:BX -> EBX (unix386.s)
815      <1> ;
816      <1> DSK_WRITE:
817      <1>      mov     ax, 0C54Ah          ; AX = NEC COMMAND, DMA COMMAND
818 00004385 66B84AC5 <1>      or      byte [MOTOR_STATUS], 10000000b ; INDICATE WRITE OPERATION
819 00004389 800D[26770100]80 <1>      ;call    RD_WR_VF          ; COMMON READ/WRITE/VERIFY
820      <1>      ;retn
821      <1>      ; 06/08/2022
822      <1>      jmp     short RD_WR_VF
823      <1>
824      <1>      ; 09/08/2022
825      <1>      ; 06/08/2022 - TRDOS 386 kernel v2.0.5
826      <1> ; -----
827      <1> ; RD_WR_VF
828      <1> ; COMMON READ, WRITE AND VERIFY:
829      <1> ; MAIN LOOP FOR STATE RETRIES.
830      <1> ;
831      <1> ; ON ENTRY: AH = READ/WRITE/VERIFY NEC PARAMETER
832      <1> ; AL = READ/WRITE/VERIFY DMA PARAMETER
833      <1> ;
834      <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
835      <1> ; -----
836      <1> ; RD_WR_VF:
837      <1>      ; 02/12/2023
838      <1>      ; 11/08/2022
839      <1>      ; 09/08/2022
840      <1>      ; 07/08/2022
841      <1>      ; 06/08/2022
842      <1>      ; 11/04/2021 (32 bit push/pop, AX -> EAX)
843      <1> ;
844      <1>
845 00004390 50          <1>      push    eax          ; SAVE DMA, NEC PARAMETERS
846      <1>
847      <1>      ; 02/12/2023
848      <1>      ; (diskette change check for 'dir', 28 seconds)
849 00004391 A0[21770100] <1>      mov     al, [TIMER_LOW+1]
850 00004396 D0E8          <1>      shr     al, 1
851 00004398 A2[867E0100] <1>      mov     [P_TIMER], al
852      <1>
853 0000439D E803040000 <1>      call    XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
854 000043A2 E86A040000 <1>      call    SETUP_STATE      ; INITIALIZE START AND END RATE
855 000043A7 58          <1>      pop     eax          ; RESTORE READ/WRITE/VERIFY
856      <1> DO_AGAIN:
857 000043A8 50          <1>      push    eax          ; SAVE READ/WRITE/VERIFY PARAMETER

```

```

858 000043A9 E8F8040000 <1> call MED_CHANGE ; MEDIA CHANGE AND RESET IF CHANGED
859 000043AE 58 <1> pop eax ; RESTORE READ/WRITE/VERIFY
860 <1> ;jc short RWV_END ; MEDIA CHANGE ERROR OR TIME-OUT
861 <1> ; 07/08/2022
862 000043AF 7305 <1> jnc short RWV
863 000043B1 E9BC000000 <1> jmp RWV_END
864 <1> RWV:
865 000043B6 50 <1> push eax ; SAVE READ/WRITE/VERIFY PARAMETER
866 000043B7 8AB7[33770100] <1> mov dh, [DSK_STATE+edi] ; GET RATE STATE OF THIS DRIVE
867 000043BD 80E6C0 <1> and dh, RATE_MSK ; KEEP ONLY RATE
868 000043C0 E83E080000 <1> call CMOS_TYPE ; RETURN DRIVE TYPE IN AL (AL)
869 <1> ; 20/02/2015
870 <1> ;jc short RWV_ASSUME ; ERROR IN CMOS
871 000043C5 7447 <1> jz short RWV_ASSUME ; 20/02/2015
872 000043C7 3C01 <1> cmp al, 1 ; 40 TRACK DRIVE?
873 000043C9 750D <1> jne short RWV_1 ; NO, BYPASS CMOS VALIDITY CHECK
874 000043CB F687[33770100]01 <1> test byte [DSK_STATE+edi], TRK_CAPA ; CHECK FOR 40 TRACK DRIVE
875 000043D2 7411 <1> jz short RWV_2 ; YES, CMOS IS CORRECT
876 <1> ;mov al, 2 ; CHANGE TO 1.2M
877 <1> ; 06/08/2022
878 000043D4 FEC0 <1> inc al ; al = 2
879 000043D6 EB0D <1> jmp short RWV_2
880 <1> RWV_1:
881 <1> ; 09/08/2022
882 <1> ;jb short RWV_2 ; NO DRIVE SPECIFIED, CONTINUE
883 000043D8 F687[33770100]01 <1> test byte [DSK_STATE+edi], TRK_CAPA ; IS IT REALLY 40 TRACK?
884 000043DF 7504 <1> jnz short RWV_2 ; NO, 80 TRACK
885 000043E1 B001 <1> mov al, 1 ; IT IS 40 TRACK, FIX CMOS VALUE
886 000043E3 EB00 <1> jmp short rwv_3
887 <1> RWV_2:
888 <1> ; 09/08/2022
889 <1> ;or al, al ; TEST FOR NO DRIVE
890 <1> ;jz short RWV_ASSUME ; ASSUME TYPE, USE MAX TRACK
891 <1> rwv_3:
892 000043E5 E855030000 <1> call DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL.
893 000043EA 7222 <1> jc short RWV_ASSUME ; TYPE NOT IN TABLE (BAD CMOS)
894 <1>
895 <1> ;----- SEARCH FOR MEDIA/DRIVE PARAMETER TABLE
896 <1>
897 000043EC 57 <1> push edi ; SAVE DRIVE #
898 <1> ; 09/08/2022
899 <1> ;xor ebx, ebx ; EBX = INDEX TO DR_TYPE TABLE
900 000043ED BB[F8650000] <1> mov ebx, DR_TYPE
901 <1> ;mov ecx, DR_CNT ; ECX = LOOP COUNT
902 000043F2 B106 <1> mov cl, DR_CNT
903 <1> RWV_DR_SEARCH:
904 <1> ;mov ah, [DR_TYPE+ebx] ; GET DRIVE TYPE
905 000043F4 8A23 <1> mov ah, [ebx]
906 000043F6 80E47F <1> and ah, BIT70FF ; MASK OUT MSB
907 000043F9 38E0 <1> cmp al, ah ; DRIVE TYPE MATCH?
908 <1> ; 09/08/2022
909 <1> ;cmp dl, ah
910 000043FB 7509 <1> jne short RWV_NXT_MD ; NO, CHECK NEXT DRIVE TYPE
911 <1> RWV_DR_FND:
912 <1> ;mov edi, [DR_TYPE+ebx+1] ; EDI = MEDIA/DRIVE PARAMETER TABLE
913 000043FD 43 <1> inc ebx
914 000043FE 8B3B <1> mov edi, [ebx]
915 00004400 4B <1> dec ebx
916 <1> RWV_MD_SEARH:
917 00004401 3A770C <1> cmp dh, [edi+MD.RATE] ; MATCH?
918 00004404 741D <1> je short RWV_MD_FND ; YES, GO GET 1ST SPECIFY BYTE
919 <1> RWV_NXT_MD:
920 00004406 83C305 <1> add ebx, 5 ; CHECK NEXT DRIVE TYPE
921 <1> ;loop RWV_DR_SEARCH
922 00004409 FEC9 <1> dec cl
923 0000440B 75E7 <1> jnz short RWV_DR_SEARCH
924 0000440D 5F <1> pop edi ; RESTORE DRIVE #
925 <1>
926 <1> ;----- ASSUME PRIMARY DRIVE IS INSTALLED AS SHIPPED
927 <1>
928 <1> RWV_ASSUME:
929 0000440E BB[16660000] <1> mov ebx, MD_TBL1 ; POINT TO 40 TRACK 250 KBS
930 00004413 F687[33770100]01 <1> test byte [DSK_STATE+edi], TRK_CAPA ; TEST FOR 80 TRACK
931 0000441A 740A <1> jz short RWV_MD_FND1 ; MUST BE 40 TRACK
932 0000441C BB[30660000] <1> mov ebx, MD_TBL3 ; POINT TO 80 TRACK 500 KBS
933 00004421 EB03 <1> jmp short RWV_MD_FND1 ; GO SPECIFY PARAMTERS
934 <1>
935 <1> ;----- CS:BX POINTS TO MEDIA/DRIVE PARAMETER TABLE
936 <1>
937 <1> RWV_MD_FND:
938 00004423 89FB <1> mov ebx, edi ; BX = MEDIA/DRIVE PARAMETER TABLE
939 00004425 5F <1> pop edi ; RESTORE DRIVE #
940 <1>
941 <1> ;----- SEND THE SPECIFY COMMAND TO THE CONTROLLER
942 <1>
943 <1> RWV_MD_FND1:
944 00004426 E85A030000 <1> call SEND_SPEC_MD
945 0000442B E8E3040000 <1> call CHK_LASTRATE ; ZF=1 ATTEMP RATE IS SAME AS LAST RATE
946 00004430 7405 <1> jz short RWV_DBL ; YES, SKIP SEND RATE COMMAND
947 00004432 E8BC040000 <1> call SEND_RATE ; SEND DATA RATE TO NEC
948 <1> RWV_DBL:
949 00004437 53 <1> push ebx ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
950 00004438 E825070000 <1> call SETUP_DBL ; CHECK FOR DOUBLE STEP
951 0000443D 5B <1> pop ebx ; RESTORE ADDRESS
952 0000443E 7225 <1> jc short CHK_RET ; ERROR FROM READ ID, POSSIBLE RETRY
953 <1> ;pop eax ; RESTORE NEC COMMAND
954 <1> ;push eax ; SAVE NEC COMMAND
955 <1> ; 09/08/2022
956 00004440 8B0424 <1> mov eax, [esp]
957 <1> ;push ebx ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
958 00004443 E8E0040000 <1> call DMA_SETUP ; SET UP THE DMA
959 <1> ;pop ebx
960 00004448 58 <1> pop eax ; RESTORE NEC COMMAND
961 00004449 7231 <1> jc short RWV_BAC ; CHECK FOR DMA BOUNDARY ERROR
962 0000444B 50 <1> push eax ; SAVE NEC COMMAND
963 0000444C 53 <1> push ebx ; SAVE MEDIA/DRIVE PARAM TBL ADDRESS
964 <1> ; 11/08/2022
965 0000444D 8B5C2420 <1> mov ebx, [esp+32] ; ECX
966 00004451 E866050000 <1> call NEC_INIT ; INITIALIZE NEC
967 00004456 5B <1> pop ebx ; RESTORE ADDRESS
968 00004457 720C <1> jc short CHK_RET ; ERROR - EXIT
969 00004459 E88C050000 <1> call RWV_COM ; OP CODE COMMON TO READ/WRITE
970 0000445E 7205 <1> jc short CHK_RET ; ERROR - EXIT
971 00004460 E8D2050000 <1> call NEC_TERM ; TERMINATE, GET STATUS, ETC.
972 <1> CHK_RET:
973 00004465 E871060000 <1> call RETRY ; CHECK FOR, SETUP RETRY
974 0000446A 58 <1> pop eax ; RESTORE READ/WRITE PARAMETER
975 0000446B 7305 <1> jnc short RWV_END ; CY = 0 NO RETRY
976 0000446D E936FFFFFF <1> jmp DO_AGAIN ; CY = 1 MEANS RETRY
977 <1> RWV_END:
978 00004472 E81C060000 <1> call DSTATE ; ESTABLISH STATE IF SUCCESSFUL
979 00004477 E8AE060000 <1> call NUM_TRANS ; AL = NUMBER TRANSFERRED
980 <1> RWV_BAC:
981 <1> ; 06/08/2022

```

```

982      <1>      ;push    eax                ; SAVE NUMBER TRANSFERRED
983      <1>      ;call    XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
984      <1>      ;pop     eax                ; RESTORE NUMBER TRANSFERRED
985      <1>      ;call    SETUP_END          ; VARIOUS CLEANUPS
986      <1>      ;retn
987      <1>
988      <1>
989      <1>      SETUP_END
990      <1>      RESTORES @MOTOR_COUNT TO PARAMETER PROVIDED IN TABLE
991      <1>      AND LOADS @DSKETTE_STATUS TO AH, AND SETS CY.
992      <1>
993      <1>      ON EXIT:
994      <1>      AH, @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
995      <1>
996      <1>      SETUP_END:
997      <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
998      <1>      ;mov     dl, 2                ; GET THE MOTOR WAIT PARAMETER
999      <1>      ;push    ax                ; SAVE NUMBER TRANSFERRED
1000     <1>      ; 11/04/2021
1001     0000447C 50      <1>      push    eax
1002     <1>      ; 06/08/2022
1003     0000447D B002    <1>      mov     al, 2                ; GET THE MOTOR WAIT PARAMETER
1004     0000447F E888070000 <1>      call    GET_PARM
1005     00004484 8825[27770100] <1>      mov     [MOTOR_COUNT], ah    ; STORE UPON RETURN
1006     <1>      ;pop     ax                ; RESTORE NUMBER TRANSFERRED
1007     <1>      ; 11/04/2021
1008     0000448A 58      <1>      pop     eax
1009     0000448B 8A25[28770100] <1>      mov     ah, [DSKETTE_STATUS] ; GET STATUS OF OPERATION
1010     00004491 08E4    <1>      or      ah, ah                ; CHECK FOR ERROR
1011     00004493 7403    <1>      jz     short NUN_ERR          ; NO ERROR
1012     00004495 30C0    <1>      xor     al, al                ; CLEAR NUMBER RETURNED
1013     <1>      ; 06/08/2022
1014     00004497 F9      <1>      stc
1015     <1>      NUN_ERR:
1016     <1>      ;cmp     ah, 1                ; SET THE CARRY FLAG TO INDICATE
1017     <1>      ;cmc
1018     00004498 C3      <1>      ; SUCCESS OR FAILURE
1019     <1>      ;retn
1020     <1>
1021     <1>      DISK_VRF (AH = 04H)
1022     <1>      DISKETTE VERIFY.
1023     <1>
1024     <1>      ON ENTRY: DI      : DRIVE #
1025     <1>      SI-HI      : HEAD #
1026     <1>      SI-LOW     : # OF SECTORS
1027     <1>      ES        : BUFFER SEGMENT
1028     <1>      [BP]       : SECTOR #
1029     <1>      [BP+1]     : TRACK #
1030     <1>      [BP+2]     : BUFFER OFFSET
1031     <1>
1032     <1>      ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1033     <1>
1034     <1>      DSK_VRF:
1035     00004499 8025[26770100]7F <1>      and     byte [MOTOR_STATUS], 01111111b ; INDICATE A READ OPERATION
1036     000044A0 66B842E6 <1>      mov     ax, 0E642h          ; AX = NEC COMMAND, DMA COMMAND
1037     <1>      ;call    RD_WR_VF          ; COMMON READ/WRITE/VERIFY
1038     <1>      ;retn
1039     <1>      ; 06/08/2022
1040     000044A4 E9E7FEFFFF <1>      jmp     RD_WR_VF
1041     <1>
1042     <1>
1043     <1>      DISK_FORMAT (AH = 05H)
1044     <1>      DISKETTE FORMAT.
1045     <1>
1046     <1>      ON ENTRY: DI      : DRIVE #
1047     <1>      SI-HI      : HEAD #
1048     <1>      SI-LOW     : # OF SECTORS
1049     <1>      ES        : BUFFER SEGMENT
1050     <1>      [BP]       : SECTOR #
1051     <1>      [BP+1]     : TRACK #
1052     <1>      [BP+2]     : BUFFER OFFSET
1053     <1>      @DISK_POINTER POINTS TO THE PARAMETER TABLE OF THIS DRIVE
1054     <1>
1055     <1>      ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1056     <1>
1057     <1>      DSK_FORMAT:
1058     <1>      ; 11/08/2022
1059     <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
1060     000044A9 E8F7020000 <1>      call    XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH.
1061     000044AE E89F030000 <1>      call    FMT_INIT          ; ESTABLISH STATE IF UNESTABLISHED
1062     000044B3 800D[26770100]80 <1>      or      byte [MOTOR_STATUS], 10000000b ; INDICATE WRITE OPERATION
1063     000044BA E8E7030000 <1>      call    MED_CHANGE        ; CHECK MEDIA CHANGE AND RESET IF SO
1064     000044BF 7261    <1>      jc      short FM_DON          ; MEDIA CHANGED, SKIP
1065     000044C1 E896020000 <1>      call    SEND_SPEC          ; SEND SPECIFY COMMAND TO NEC
1066     000044C6 E848040000 <1>      call    CHK_LASTRATE        ; ZF=1 ATTEMPT RATE IS SAME AS LAST RATE
1067     000044CB 7405    <1>      jz     short FM_WR          ; YES, SKIP SPECIFY COMMAND
1068     000044CD E821040000 <1>      call    SEND_RATE          ; SEND DATA RATE TO CONTROLLER
1069     <1>      FM_WR:
1070     000044D2 E8C8040000 <1>      call    FMTDMA_SET          ; SET UP THE DMA FOR FORMAT
1071     000044D7 7249    <1>      jc      short FM_DON          ; RETURN WITH ERROR
1072     000044D9 B44D    <1>      mov     ah, 04Dh          ; ESTABLISH THE FORMAT COMMAND
1073     <1>      ; 11/08/2022
1074     000044DB 8B5C2418 <1>      mov     ebx, [esp+24] ; ECX
1075     000044DF E8D8040000 <1>      call    NEC_INIT          ; INITIALIZE THE NEC
1076     000044E4 723C    <1>      jc      short FM_DON          ; ERROR - EXIT
1077     000044E6 B8[22450000] <1>      mov     eax, FM_DON          ; LOAD ERROR ADDRESS
1078     000044EB 50      <1>      push    eax                ; PUSH NEC_OUT ERROR RETURN
1079     <1>      ;mov     dl, 3                ; BYTES/SECTOR VALUE TO NEC
1080     <1>      ; 06/08/2022
1081     000044EC B003    <1>      mov     al, 3
1082     000044EE E819070000 <1>      call    GET_PARM
1083     000044F3 E80A080000 <1>      call    NEC_OUTPUT          ; SECTORS/TRACK VALUE TO NEC
1084     <1>      ;mov     dl, 4
1085     <1>      ; 06/08/2022
1086     000044F8 B004    <1>      mov     al, 4
1087     000044FA E80D070000 <1>      call    GET_PARM
1088     000044FF E8FE070000 <1>      call    NEC_OUTPUT          ; GAP LENGTH VALUE TO NEC
1089     <1>      ;mov     dl, 7
1090     <1>      ; 06/08/2022
1091     00004504 B007    <1>      mov     al, 7
1092     00004506 E801070000 <1>      call    GET_PARM
1093     0000450B E8F2070000 <1>      call    NEC_OUTPUT          ; FILLER BYTE TO NEC
1094     <1>      ;mov     dl, 8
1095     <1>      ; 06/08/2022
1096     00004510 B008    <1>      mov     al, 8
1097     00004512 E8F5060000 <1>      call    GET_PARM
1098     00004517 E8E6070000 <1>      call    NEC_OUTPUT
1099     0000451C 58      <1>      pop     eax                ; THROW AWAY ERROR
1100     0000451D E815050000 <1>      call    NEC_TERM          ; TERMINATE, RECEIVE STATUS, ETC,
1101     <1>      FM_DON:
1102     <1>      ; 06/08/2022
1103     <1>      ;call    XLAT_OLD          ; TRANSLATE STATE TO COMPATIBLE MODE
1104     <1>      ; 06/08/2022
1105     00004522 E947FEFFFF <1>      jmp     SETUP_END_X

```

```

1106      <1>      ;call SETUP_END          ; VARIOUS CLEANUPS
1107      <1>      ;; 06/08/2022
1108      <1>      ;mov ebx, esi          ; GET SAVED AL TO BL
1109      <1>      ;mov al, bl            ; PUT BACK FOR RETURN
1110      <1>      ;retn
1111      <1>
1112      <1>      -----
1113      <1>      FNC_ERR
1114      <1>      ; INVALID FUNCTION REQUESTED OR INVALID DRIVE:
1115      <1>      ; SET BAD COMMAND IN STATUS.
1116      <1>
1117      <1>      ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
1118      <1>      -----
1119      <1>      FNC_ERR:                ; INVALID FUNCTION REQUEST
1120      <1>      ;mov ax, si
1121      <1>      ; 06/08/2022
1122      <1>      mov eax, esi          ; RESTORE AL
1123      <1>      mov ah, BAD_CMD       ; SET BAD COMMAND ERROR
1124      <1>      mov [DSKETTE_STATUS], ah ; STORE IN DATA AREA
1125      <1>      stc                   ; SET CARRY INDICATING ERROR
1126      <1>      retn
1127      <1>
1128      <1>      ; 06/08/2022 - TRDOS 386 Kernel v2.0.5
1129      <1>      ; 30/08/2020
1130      <1>      ; 29/08/2020
1131      <1>      ; 01/06/2016
1132      <1>      ; 28/05/2016
1133      <1>      ; 27/05/2016 - TRDOS 386 (TRDOS v.2.0)
1134      <1>      -----
1135      <1>      DISK_PARAMS (AH = 08H)
1136      <1>      ; READ DRIVE PARAMETERS.
1137      <1>
1138      <1>      ON ENTRY: DI : DRIVE #
1139      <1>      ; 27/05/2016
1140      <1>      ; EBX = Buffer Address for floppy disk parameters table (16 bytes)
1141      <1>
1142      <1>      ON EXIT: CL/[BP] = BITS 7 & 6 HI 2 BITS OF MAX CYLINDER
1143      <1>      ; BITS 0-5 MAX SECTORS/TRACK
1144      <1>      ; CH/[BP+1] = LOW 8 BITS OF MAX CYLINDER
1145      <1>      ; BL/[BP+2] = BITS 7-4 = 0
1146      <1>      ; BITS 3-0 = VALID CMOS DRIVE TYPE
1147      <1>      ; BH/[BP+3] = 0
1148      <1>      ; DL/[BP+4] = # DRIVES INSTALLED (VALUE CHECKED)
1149      <1>      ; DH/[BP+5] = MAX HEAD #
1150      <1>      ; ** 27/05/2016 - TRDOS 386 (TRDOS v2.0) **
1151      <1>      ; ** EBX = Buffer address for floppy disk parameters table **
1152      <1>      ; DI/[BP+6] = OFFSET TO DISK_BASE
1153      <1>      ; ES = SEGMENT OF DISK_BASE
1154      <1>
1155      <1>      ; AX = 0
1156      <1>
1157      <1>      ; NOTE : THE ABOVE INFORMATION IS STORED IN THE USERS STACK AT
1158      <1>      ; THE LOCATIONS WHERE THE MAIN ROUTINE WILL POP THEM
1159      <1>      ; INTO THE APPROPRIATE REGISTERS BEFORE RETURNING TO THE
1160      <1>      ; CALLER.
1161      <1>      -----
1162      <1>      DSK_PARAMS:
1163      <1>      ; 09/08/2022
1164      <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
1165      <1>
1166      <1>      ; Registers on stack:
1167      <1>      ; ebx = esp+28
1168      <1>      ; ecx = esp+24
1169      <1>      ; edx = esp+20
1170      <1>      ; esi = esp+16
1171      <1>      ; edi = esp+12
1172      <1>      ; return address from DSKETTE_IO_1 = esp+8
1173      <1>      ; ebp = esp+4
1174      <1>      ; return address from DSK_PARAMS = esp
1175      <1>
1176      <1>      ; INPUT:
1177      <1>      ; ebp = buffer address
1178      <1>      ; edi = drive number (0 or 1)
1179      <1>
1180      <1>      ; OUTPUT:
1181      <1>      ; ebx = [esp+28] ((BL = cmos type, BH = 0))
1182      <1>      ; ecx = [esp+24] ((CL = sectors per track, CH = tracks - 1))
1183      <1>      ; edx = [esp+20] ((DL = floppy drive count, DH = heads - 1))
1184      <1>      ; user's buffer = FDPT table
1185      <1>
1186      <1>      call XLAT_NEW          ; TRANSLATE STATE TO PRESENT ARCH,
1187      <1>
1188      <1>      ;;mov word [bp+2], 0      ; DRIVE TYPE = 0
1189      <1>      ;;mov ax, [EQUIP_FLAG]   ; LOAD EQUIPMENT FLAG FOR # DISKETTES
1190      <1>      ;;and al, 11000001b      ; KEEP DISKETTE DRIVE BITS
1191      <1>      ;;mov dl, 2              ; DISKETTE DRIVES = 2
1192      <1>      ;;cmp al, 01000001b      ; 2 DRIVES INSTALLED ?
1193      <1>      ;;jz short STO_DL         ; IF YES JUMP
1194      <1>      ;;dec dl                 ; DISKETTE DRIVES = 1
1195      <1>      ;;cmp al, 00000001b      ; 1 DRIVE INSTALLED ?
1196      <1>      ;;jnz short NON_DRV      ; IF NO JUMP
1197      <1>
1198      <1>      sub edx, edx
1199      <1>      mov ax, [fd0_type]
1200      <1>      and ax, ax
1201      <1>      jz short NON_DRV
1202      <1>      inc dl
1203      <1>      and ah, ah
1204      <1>      jz short STO_DL
1205      <1>      inc dl
1206      <1>      STO_DL:
1207      <1>      ; 30/08/2020
1208      <1>      ; cmp dx, di
1209      <1>      ; 06/08/2022
1210      <1>      cmp edx, edi
1211      <1>      jna short NON_DRV
1212      <1>      ;
1213      <1>      ; mov [bp+4], dl          ; STORE NUMBER OF DRIVES
1214      <1>      ; mov [ebp+8], edx ; 20/02/2015
1215      <1>
1216      <1>      ; 06/08/2022
1217      <1>      inc dh ; number of heads - 1
1218      <1>      ; dh = 1
1219      <1>      ; dl = number of floppy/diskette drives (DL)
1220      <1>      mov [esp+20], edx
1221      <1>
1222      <1>      ; 11/04/2021
1223      <1>      ; cmp di, 1            ; CHECK FOR VALID DRIVE
1224      <1>      ; ja short NON_DRV1     ; DRIVE INVALID
1225      <1>      ; ja NON_DRV1 ; 29/08/2020
1226      <1>      ;
1227      <1>      ; mov byte [bp+5], 1     ; MAXIMUM HEAD NUMBER = 1
1228      <1>      ; 06/08/2022
1229      <1>      ; mov byte [ebp+9], 1 ; 20/02/2015

```

```

1230 <1>
1231 00004557 E8A7060000 <1> call CMOS_TYPE ; RETURN DRIVE TYPE IN AL
1232 <1> ; 20/02/2015
1233 <1> ;jc short CHK_EST ; IF CMOS BAD CHECKSUM ESTABLISHED
1234 <1> ;or al, al ; TEST FOR NO DRIVE TYPE
1235 0000455C 740F <1> jz short CHK_EST ; JUMP IF SO
1236 0000455E E8DC010000 <1> call DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1237 00004563 7208 <1> jc short CHK_EST ; TYPE NOT IN TABLE (POSSIBLE BAD CMOS)
1238 <1> ;mov [bp+2], al ; STORE VALID CMOS DRIVE TYPE
1239 <1> ;mov [ebp+4], al ; 06/02/2015
1240 00004565 8A4B04 <1> mov cl, [ebx+MD.SEC_TRK] ; GET SECTOR/TRACK
1241 00004568 8A6B0B <1> mov ch, [ebx+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
1242 0000456B EB36 <1> jmp short STO_CX ; CMOS GOOD, USE CMOS
1243 <1> CHK_EST:
1244 0000456D 8AA7[33770100] <1> mov ah, [DSK_STATE+edi] ; LOAD STATE FOR THIS DRIVE
1245 00004573 F6C410 <1> test ah, MED_DET ; CHECK FOR ESTABLISHED STATE
1246 00004576 744D <1> jz short NON_DRV1 ; CMOS BAD/INVALID OR UNESTABLISHED
1247 <1> USE_EST:
1248 00004578 80E4C0 <1> and ah, RATE_MSK ; ISOLATE STATE
1249 0000457B 80FC80 <1> cmp ah, RATE_250 ; RATE 250 ?
1250 0000457E 755C <1> jne short USE_EST2 ; NO, GO CHECK OTHER RATE
1251 <1>
1252 <1> ;----- DATA RATE IS 250 KBS, TRY 360 KB TABLE FIRST
1253 <1>
1254 00004580 B001 <1> mov al, 1 ; DRIVE TYPE 1 (360KB)
1255 00004582 E8B8010000 <1> call DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1256 00004587 8A4B04 <1> mov cl, [ebx+MD.SEC_TRK] ; GET SECTOR/TRACK
1257 0000458A 8A6B0B <1> mov ch, [ebx+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
1258 0000458D F687[33770100]01 <1> test byte [DSK_STATE+edi], TRK_CAPA ; 80 TRACK ?
1259 00004594 740D <1> jz short STO_CX ; MUST BE 360KB DRIVE
1260 <1>
1261 <1> ;----- IT IS 1.44 MB DRIVE
1262 <1>
1263 <1> PARM144:
1264 00004596 B004 <1> mov al, 4 ; DRIVE TYPE 4 (1.44MB)
1265 00004598 E8A2010000 <1> call DR_TYPE_CHECK ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1266 0000459D 8A4B04 <1> mov cl, [ebx+MD.SEC_TRK] ; GET SECTOR/TRACK
1267 000045A0 8A6B0B <1> mov ch, [ebx+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
1268 <1> STO_CX:
1269 <1> ;mov [ebp], ecx ; SAVE POINTER IN STACK FOR RETURN
1270 <1> ; 06/08/2022
1271 000045A3 894C2418 <1> mov [esp+24], ecx ; spt (cl), tracks - 1 (ch)
1272 <1> ES_DI:
1273 <1> ;mov [bp+6], bx ; ADDRESS OF MEDIA/DRIVE PARM TABLE
1274 <1> ;mov [ebp+12], ebx ; 06/02/2015
1275 <1> ;mov ax, cs ; SEGMENT MEDIA/DRIVE PARAMETER TABLE
1276 <1> ;mov es, ax ; ES IS SEGMENT OF TABLE
1277 <1>
1278 <1> ; 28/05/2016
1279 <1> ; 27/05/2016
1280 <1> ; return floppy disk parameters table to user
1281 <1> ; in user's buffer, which is pointed by EBX
1282 <1>
1283 <1> ; 09/08/2022
1284 <1> ;movzx eax, al
1285 000045A7 29C9 <1> sub ecx, ecx
1286 000045A9 88C1 <1> mov cl, al
1287 <1> ;mov [ebp+4], eax ; ebx ; drive type (for floppy drives)
1288 <1> ; 06/08/2022
1289 <1> ;mov [esp+28], eax ; drive type
1290 <1> ; 09/08/2022
1291 000045AB 894C241C <1> mov [esp+28], ecx ; drive type
1292 <1>
1293 <1> ; 06/08/2022
1294 <1> ;push edi
1295 <1>
1296 <1> ;mov edi, [ebp+4] ; ebx (input), user's buffer address
1297 <1> ; 06/08/2022
1298 <1> ;mov edi, ebp ; [esp+28] ; user's buffer address
1299 <1>
1300 <1> ; 29/08/2020
1301 <1> ;or edi, edi
1302 <1> ;jz short no_copy_fdpt
1303 <1> ; 06/08/2022
1304 000045AF 09ED <1> or ebp, ebp ; [esp+28] = ebx ; user's buffer address if > 0
1305 000045B1 740B <1> jz short DP_OUT
1306 <1> ; 06/08/2022
1307 <1> ;push edi
1308 000045B3 89EF <1> mov edi, ebp
1309 <1>
1310 <1> ; 06/08/2022
1311 <1> ; 01/06/2016 (Int 33h, disk type return for floppy disks, in BL)
1312 <1> ;mov [user_buffer], eax ; 01/06/2016 (overwrite ebx return value)
1313 <1>
1314 <1> ; (Int 33h, Function 08h will replace user's buffer addr with disk type!)
1315 <1>
1316 000045B5 89DE <1> mov esi, ebx ; floppy disk parameter table (16 bytes)
1317 <1> ;mov ecx, 16 ; 16 bytes
1318 <1> ; 09/08/2022
1319 <1> ; ecx < 256
1320 <1> ; 06/08/2022
1321 <1> ;sub ecx, ecx
1322 000045B7 B110 <1> mov cl, 16
1323 000045B9 E84DC50000 <1> call transfer_to_user_buffer ; trdosk6.s (16/05/2016)
1324 <1> no_copy_fdpt:
1325 <1> ; 06/08/2022
1326 <1> ;pop edi
1327 <1> DP_OUT:
1328 <1> ; 06/08/2022
1329 <1> ;call XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
1330 <1> ;xor ax, ax ; CLEAR
1331 <1> ; 06/08/2022
1332 000045BE 31C0 <1> xor eax, eax
1333 <1> ;clc
1334 000045C0 C3 <1> retn
1335 <1>
1336 <1> ;----- NO DRIVE PRESENT HANDLER
1337 <1>
1338 <1> NON_DRV:
1339 <1> ;mov byte [bp+4], 0 ; CLEAR NUMBER OF DRIVES
1340 <1> ;mov [ebp+8], edx ; 0 ; 20/02/2015
1341 <1> ; 06/08/2022
1342 000045C1 89542414 <1> mov [esp+20], edx ; 0 ; number of floppy drives and heads are 0
1343 <1> NON_DRV1:
1344 000045C5 6681FF8000 <1> di, 80h ; CHECK FOR FIXED MEDIA TYPE REQUEST
1345 000045CA 7206 <1> jnb short NON_DRV2 ; CONTINUE IF NOT REQUEST FALL THROUGH
1346 <1>
1347 <1> ;----- FIXED DISK REQUEST FALL THROUGH ERROR
1348 <1>
1349 <1> ; 06/08/2022
1350 <1> ;call XLAT_OLD ; ELSE TRANSLATE TO COMPATIBLE MODE
1351 <1> ;mov ax, si ; RESTORE AL
1352 <1> ; 06/08/2022
1353 000045CC 89F0 <1> mov eax, esi

```

```

1354 000045CE B401      <1>      mov     ah, BAD_CMD          ; SET BAD COMMAND ERROR
1355 000045D0 F9        <1>      stc
1356 000045D1 C3        <1>      retn
1357                    <1>
1358                    <1> NON_DRV2:
1359                    <1>      ;xor     ax, ax                ; CLEAR PARMS IF NO DRIVES OR CMOS BAD
1360 000045D2 31C0       <1>      xor     eax, eax
1361                    <1>      ;mov     [ebp], ax          ; TRACKS, SECTORS/TRACK = 0
1362                    <1>      ; 06/08/2022
1363 000045D4 89442418   <1>      mov     [esp+24], eax ; spt and max. track number is 0
1364                    <1>
1365                    <1>      ;mov     [bp+5], ah          ; HEAD = 0
1366                    <1>      ; 06/08/2022
1367                    <1>      ;mov     [ebp+9], ah ; 06/02/2015
1368                    <1>      ; 06/08/2022
1369                    <1>      ;mov     [esp+21], ah ; 0
1370                    <1>
1371                    <1>      ;mov     [bp+6], ax          ; OFFSET TO DISK_BASE = 0
1372                    <1>      ; 06/08/2022
1373                    <1>      ;mov     [ebp+12], eax
1374                    <1>
1375                    <1>      ;mov     es, ax                ; ES IS SEGMENT OF TABLE
1376                    <1>      ;jmp     short DP_OUT
1377                    <1>
1378                    <1>      ; 06/08/2022
1379                    <1>      ; 30/08/2020
1380                    <1>      ;call    XLAT_OLD
1381                    <1>      ;mov     ah, NOT_RDY ; drive not ready
1382 000045D8 B407       <1>      mov     ah, INIT_FAIL ; DRIVE PARAMETER ACTIVITY FAILED
1383 000045DA F9        <1>      stc
1384 000045DB C3        <1>      retn
1385                    <1>
1386                    <1> ;----- DATA RATE IS EITHER 300 KBS OR 500 KBS, TRY 1.2 MB TABLE FIRST
1387                    <1>
1388                    <1> USE_EST2:
1389                    <1>      mov     al, 2                ; DRIVE TYPE 2 (1.2MB)
1390 000045DE E85C010000 <1>      call   DR_TYPE_CHECK          ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1391 000045E3 8A4B04     <1>      mov     cl, [ebx+MD.SEC_TRK] ; GET SECTOR/TRACK
1392 000045E6 8A6B0B     <1>      mov     ch, [ebx+MD.MAX_TRK] ; GET MAX. TRACK NUMBER
1393 000045E9 80FC40     <1>      cmp     ah, RATE_300          ; RATE 300 ?
1394 000045EC 74B5      <1>      jz      short STO_CX          ; MUST BE 1.2MB DRIVE
1395 000045EE EBA6      <1>      jmp     short PARM144         ; ELSE, IT IS 1.44MB DRIVE
1396                    <1>
1397                    <1> ; 30/08/2020
1398                    <1>
1399                    <1> ;-----
1400                    <1> ; DISK_TYPE (AH = 15H)
1401                    <1> ; THIS ROUTINE RETURNS THE TYPE OF MEDIA INSTALLED.
1402                    <1> ;
1403                    <1> ; ON ENTRY: DI = DRIVE #
1404                    <1> ;
1405                    <1> ; ON EXIT: AH = DRIVE TYPE, CY=0
1406                    <1> ;-----
1407                    <1> DSK_TYPE:
1408                    <1> ; 06/08/2022 - TRDOS 386 v2.0.5
1409 000045F0 E8B0010000 <1>      call   XLAT_NEW              ; TRANSLATE STATE TO PRESENT ARCH.
1410 000045F5 8A87[33770100] <1>      mov     al, [DSK_STATE+edi] ; GET PRESENT STATE INFORMATION
1411 000045FB 08C0      <1>      or      al, al                ; CHECK FOR NO DRIVE
1412 000045FD 740D      <1>      jz      short NO_DRV          ; NO CHANGE LINE FOR 40 TRACK DRIVE
1413 000045FF B401      <1>      mov     ah, NOCHGLN          ; NO CHANGE LINE FOR 40 TRACK DRIVE
1414 00004601 A801      <1>      test    al, TRK_CAPA          ; IS THIS DRIVE AN 80 TRACK DRIVE?
1415 00004603 7402      <1>      jz      short DT_BACK        ; IF NO JUMP
1416 00004605 B402      <1>      mov     ah, CHGLN            ; CHANGE LINE FOR 80 TRACK DRIVE
1417                    <1> DT_BACK:
1418                    <1> ; 06/08/2022
1419                    <1> ;push    ax                ; SAVE RETURN VALUE
1420                    <1> ; 11/04/2021
1421                    <1> ;push    eax
1422                    <1> ;call    XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
1423                    <1> ; 11/04/2021
1424                    <1> ;pop     eax
1425                    <1> ;pop     ax
1426                    <1> ;clic
1427                    <1> ;mov     bx, si            ; NO ERROR
1428                    <1> ; 06/08/2022          ; GET SAVED AL TO BL
1429 00004607 89F3      <1>      mov     ebx, esi
1430 00004609 88D8      <1>      mov     al, bl
1431 0000460B C3        <1>      retn
1432                    <1> NO_DRV:
1433                    <1> ;xor     ah, ah                ; NO DRIVE PRESENT OR UNKNOWN
1434                    <1> ;jmp     short DT_BACK
1435                    <1>
1436                    <1> ; 06/08/2022
1437                    <1> ; 30/08/2020
1438                    <1> ;call    XLAT_OLD
1439 0000460C 29C0      <1>      sub     eax, eax
1440 0000460E F9        <1>      stc ; cf = 1 -> drive not ready, ah = 0 (disk type = 0)
1441 0000460F C3        <1>      retn
1442                    <1>
1443                    <1> ;-----
1444                    <1> ; DISK_CHANGE (AH = 16H)
1445                    <1> ; THIS ROUTINE RETURNS THE STATE OF THE DISK CHANGE LINE.
1446                    <1> ;
1447                    <1> ; ON ENTRY: DI = DRIVE #
1448                    <1> ;
1449                    <1> ; ON EXIT: AH = @DSKETTE_STATUS
1450                    <1> ; 00 - DISK CHANGE LINE INACTIVE, CY = 0
1451                    <1> ; 06 - DISK CHANGE LINE ACTIVE, CY = 1
1452                    <1> ;-----
1453                    <1> DSK_CHANGE:
1454                    <1> ; 06/08/2022 - TRDOS 386 v2.0.5
1455 00004610 E890010000 <1>      call   XLAT_NEW              ; TRANSLATE STATE TO PRESENT ARCH.
1456 00004615 8A87[33770100] <1>      mov     al, [DSK_STATE+edi] ; GET MEDIA STATE INFORMATION
1457 0000461B 08C0      <1>      or      al, al                ; DRIVE PRESENT ?
1458 0000461D 7417      <1>      jz      short DC_NON          ; JUMP IF NO DRIVE
1459 0000461F A801      <1>      test    al, TRK_CAPA          ; 80 TRACK DRIVE ?
1460 00004621 7407      <1>      jz      short SETIT          ; IF SO , CHECK CHANGE LINE
1461                    <1> DC0:
1462 00004623 E86B080000 <1>      call   READ_DSKCHNG          ; GO CHECK STATE OF DISK CHANGE LINE
1463 00004628 7407      <1>      jz      short FINIS          ; CHANGE LINE NOT ACTIVE
1464                    <1> SETIT:
1465 0000462A C605[28770100]06 <1>      mov     byte [DSKETTE_STATUS], MEDIA_CHANGE ; INDICATE MEDIA REMOVED
1466                    <1>
1467                    <1> FINIS: ; 06/08/2022
1468                    <1> ;call    XLAT_OLD            ; TRANSLATE STATE TO COMPATIBLE MODE
1469                    <1> ; 06/08/2022
1470 00004631 E938FDFFFF <1>      jmp     SETUP_END_X          ; VARIOUS CLEANUPS
1471                    <1> ;call    SETUP_END
1472                    <1> ; 06/08/2022
1473                    <1> ;mov     ebx, esi            ; GET SAVED AL TO BL
1474                    <1> ;mov     al, bl              ; PUT BACK FOR RETURN
1475                    <1> ;retn
1476                    <1> DC_NON:
1477 00004636 800D[28770100]80 <1>      or      byte [DSKETTE_STATUS], TIME_OUT ; SET TIMEOUT, NO DRIVE

```

```

1478 0000463D EBF2      <1>      jmp      short FINIS
1479                  <1>
1480                  <1> -----
1481                  <1>      FORMAT_SET      (AH = 17H)
1482                  <1>      THIS ROUTINE IS USED TO ESTABLISH THE TYPE OF MEDIA TO BE USED
1483                  <1>      FOR THE FOLLOWING FORMAT OPERATION.
1484                  <1>
1485                  <1>      ON ENTRY: SI LOW = DASD TYPE FOR FORMAT
1486                  <1>      DI      = DRIVE #
1487                  <1>
1488                  <1>      ON EXIT: @DSKETTE_STATUS REFLECTS STATUS
1489                  <1>      AH = @DSKETTE_STATUS
1490                  <1>      CY = 1 IF ERROR
1491                  <1> -----
1492                  <1>      FORMAT_SET:
1493 0000463F E861010000  <1>      call     XLAT_NEW      ; TRANSLATE STATE TO PRESENT ARCH.
1494                  <1>      ;push     si      ; SAVE DASD TYPE
1495                  <1>      ; 11/04/2021
1496 00004644 56          <1>      push     esi
1497                  <1>      ;mov     ax, si      ; AH = ? , AL = DASD TYPE
1498                  <1>      ;xor     ah, ah      ; AH = 0 , AL = DASD TYPE
1499                  <1>      ;mov     si, ax      ; SI = DASD TYPE
1500                  <1>      ; 11/04/2021
1501 00004645 89F0        <1>      mov     eax, esi
1502 00004647 0FB6F0      <1>      movzx   esi, al
1503 0000464A 80A7[33770100]0F <1>      and     byte [DSK_STATE+edi], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
1504                  <1>      ;dec     si      ; CHECK FOR 320/360K MEDIA & DRIVE
1505                  <1>      ; 11/04/2021
1506 00004651 4E          <1>      dec     esi
1507 00004652 7509        <1>      jnz     short NOT_320      ; BYPASS IF NOT
1508 00004654 808F[33770100]90 <1>      or      byte [DSK_STATE+edi], MED_DET+RATE_250 ; SET TO 320/360
1509 0000465B EB45        <1>      jmp     short S0
1510                  <1>
1511                  <1>      NOT_320:
1512 0000465D E844020000  <1>      call     MED_CHANGE      ; CHECK FOR TIME_OUT
1513 00004662 803D[28770100]80 <1>      cmp     byte [DSKETTE_STATUS], TIME_OUT
1514 00004669 7437        <1>      jz      short S0      ; IF TIME OUT TELL CALLER
1515                  <1>
1516                  <1>      S3:
1517                  <1>      ;dec     si      ; CHECK FOR 320/360K IN 1.2M DRIVE
1518                  <1>      ; 11/04/2021
1519 0000466B 4E          <1>      dec     esi
1520 0000466C 7509        <1>      jnz     short NOT_320_12      ; BYPASS IF NOT
1521 0000466E 808F[33770100]70 <1>      or      byte [DSK_STATE+edi], MED_DET+DBL_STEP+RATE_300 ; SET STATE
1522 00004675 EB2B        <1>      jmp     short S0
1523                  <1>
1524                  <1>      NOT_320_12:
1525                  <1>      ;dec     si      ; CHECK FOR 1.2M MEDIA IN 1.2M DRIVE
1526                  <1>      ; 11/04/2021
1527 00004677 4E          <1>      dec     esi
1528 00004678 7509        <1>      jnz     short NOT_12      ; BYPASS IF NOT
1529 0000467A 808F[33770100]10 <1>      or      byte [DSK_STATE+edi], MED_DET+RATE_500 ; SET STATE VARIABLE
1530 00004681 EB1F        <1>      jmp     short S0      ; RETURN TO CALLER
1531                  <1>
1532                  <1>      NOT_12:
1533                  <1>      ;dec     si      ; CHECK FOR SET DASD TYPE 04
1534                  <1>      ; 11/04/2021
1535 00004683 4E          <1>      dec     esi
1536 00004684 7525        <1>      jnz     short FS_ERR      ; BAD COMMAND EXIT IF NOT VALID TYPE
1537 00004686 F687[33770100]04 <1>      test    byte [DSK_STATE+edi], DRV_DET ; DRIVE DETERMINED ?
1538 0000468D 740B        <1>      jz      short ASSUME      ; IF STILL NOT DETERMINED ASSUME
1539 0000468F B050        <1>      mov     al, MED_DET+RATE_300
1540 00004691 F687[33770100]02 <1>      test    byte [DSK_STATE+edi], FMT_CAPA ; MULTIPLE FORMAT CAPABILITY ?
1541 00004698 7502        <1>      jnz     short OR_IT_IN      ; IF 1.2 M THEN DATA RATE 300
1542                  <1>
1543                  <1>      ASSUME:
1544 0000469A B090        <1>      mov     al, MED_DET+RATE_250 ; SET UP
1545                  <1>
1546                  <1>      OR_IT_IN:
1547 0000469C 0887[33770100] <1>      or      [DSK_STATE+edi], al ; OR IN THE CORRECT STATE
1548                  <1>
1549                  <1>      S0:
1550                  <1>      ; 06/08/2022
1551 000046A2 E8D5FDFFFF  <1>      call     XLAT_OLD      ; TRANSLATE STATE TO COMPATIBLE MODE
1552                  <1>      call     SETUP_END      ; VARIOUS CLEANUPS
1553                  <1>      ;pop     bx      ; GET SAVED AL TO BL
1554                  <1>      ; 11/04/2021
1555 000046A7 5B          <1>      pop     ebx
1556 000046A8 88D8        <1>      mov     al, bl      ; PUT BACK FOR RETURN
1557 000046AA C3          <1>      retn
1558                  <1>
1559 000046AB C605[28770100]01 <1>      FS_ERR:
1560 000046B2 EBEE        <1>      mov     byte [DSKETTE_STATUS], BAD_CMD ; UNKNOWN STATE,BAD COMMAND
1561                  <1>      jmp     short S0
1562                  <1> -----
1563                  <1>      SET_MEDIA (AH = 18H)
1564                  <1>      THIS ROUTINE SETS THE TYPE OF MEDIA AND DATA RATE
1565                  <1>      TO BE USED FOR THE FOLLOWING FORMAT OPERATION.
1566                  <1>
1567                  <1>      ON ENTRY:
1568                  <1>      [BP] = SECTOR PER TRACK
1569                  <1>      [BP+1] = TRACK #
1570                  <1>      DI = DRIVE #
1571                  <1>
1572                  <1>      ON EXIT:
1573                  <1>      @DSKETTE_STATUS REFLECTS STATUS
1574                  <1>      IF NO ERROR:
1575                  <1>      AH = 0
1576                  <1>      CY = 0
1577                  <1>      ES = SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
1578                  <1>      DI/[BP+6] = OFFSET OF MEDIA/DRIVE PARAMETER TABLE
1579                  <1>      IF ERROR:
1580                  <1>      AH = @DSKETTE_STATUS
1581                  <1>      CY = 1
1582                  <1> -----
1583                  <1>      SET_MEDIA:
1584                  <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
1585 000046B4 E8EC000000  <1>      call     XLAT_NEW      ; TRANSLATE STATE TO PRESENT ARCH.
1586 000046B9 F687[33770100]01 <1>      test    byte [DSK_STATE+edi], TRK_CAPA ; CHECK FOR CHANGE LINE AVAILABLE
1587 000046C0 7415        <1>      jz      short SM_CMOS      ; JUMP IF 40 TRACK DRIVE
1588 000046C2 E8DF010000  <1>      call     MED_CHANGE      ; RESET CHANGE LINE
1589 000046C7 803D[28770100]80 <1>      cmp     byte [DSKETTE_STATUS], TIME_OUT ; IF TIME OUT TELL CALLER
1590 000046CE 746A        <1>      je      short SM_RTN
1591 000046D0 C605[28770100]00 <1>      mov     byte [DSKETTE_STATUS], 0 ; CLEAR STATUS
1592                  <1>
1593 000046D7 E827050000  <1>      SM_CMOS:
1594                  <1>      call     CMOS_TYPE      ; RETURN DRIVE TYPE IN (AL)
1595                  <1>      ; 20/02/2015
1596                  <1>      ;jc      short MD_NOT_FND      ; ERROR IN CMOS
1597 000046DC 745C        <1>      ;or      al, al      ; TEST FOR NO DRIVE
1598 000046DE E85C000000  <1>      jz      short SM_RTN      ; RETURN IF SO
1599 000046E3 7233        <1>      call     DR_TYPE_CHECK      ; RTN CS:BX = MEDIA/DRIVE PARAM TBL
1600 000046E5 57          <1>      jc      short MD_NOT_FND      ; TYPE NOT IN TABLE (BAD CMOS)
1601 000046E6 31DB        <1>      push     edi      ; SAVE REG.
1601 000046E6 31DB        <1>      xor     ebx, ebx      ; BX = INDEX TO DR. TYPE TABLE

```

```

1602 000046E8 B906000000 <1> mov ecx, DR_CNT ; CX = LOOP COUNT
1603 <1> DR_SEARCH: <1>
1604 000046ED 8AA3[F8650000] <1> mov ah, [DR_TYPE+ebx] ; GET DRIVE TYPE
1605 000046F3 80E47F <1> and ah, BIT7OFF ; MASK OUT MSB
1606 000046F6 38E0 <1> cmp al, ah ; DRIVE TYPE MATCH ?
1607 000046F8 7518 <1> jne short NXT_MD ; NO, CHECK NEXT DRIVE TYPE
1608 <1> DR_FND: <1>
1609 000046FA 8BBB[F9650000] <1> mov edi, [DR_TYPE+ebx+1] ; DI = MEDIA/DRIVE PARAM TABLE
1610 <1> MD_SEARCH: <1>
1611 00004700 8A6704 <1> mov ah, [edi+MD.SEC_TRK] ; GET SECTOR/TRACK
1612 <1> ; cmp [ebp], ah ; MATCH?
1613 <1> ; 06/08/2022
1614 00004703 38642418 <1> cmp [esp+24], ah ; CL ; spt
1615 00004707 7509 <1> jne short NXT_MD ; NO, CHECK NEXT MEDIA
1616 00004709 8A670B <1> mov ah, [edi+MD.MAX_TRK] ; GET MAX. TRACK #
1617 <1> ; cmp [ebp+1], ah ; MATCH?
1618 <1> ; 06/08/2022
1619 0000470C 38642419 <1> cmp [esp+25], ah ; CH ; heads - 1
1620 00004710 740F <1> je short MD_FND ; YES, GO GET RATE
1621 <1> NXT_MD: <1>
1622 <1> ; add bx, 3 ; CHECK NEXT DRIVE TYPE
1623 00004712 83C305 <1> add ebx, 5 ; 18/02/2015
1624 00004715 E2D6 <1> loop DR_SEARCH
1625 00004717 5F <1> pop edi ; RESTORE REG.
1626 <1> MD_NOT_FND: <1>
1627 00004718 C605[28770100]0C <1> mov byte [DSKETTE_STATUS], MED_NOT_FND ; ERROR, MEDIA TYPE NOT FOUND
1628 0000471F EB19 <1> jmp short SM_RTN ; RETURN
1629 <1> MD_FND: <1>
1630 00004721 8A470C <1> mov al, [edi+MD.RATE] ; GET RATE
1631 00004724 3C40 <1> cmp al, RATE_300 ; DOUBLE STEP REQUIRED FOR RATE 300
1632 00004726 7502 <1> jne short MD_SET
1633 00004728 0C20 <1> or al, DBL_STEP
1634 <1> MD_SET: <1>
1635 <1> ; mov [bp+6], di ; SAVE TABLE POINTER IN STACK
1636 <1> ; 06/08/2022
1637 <1> ; mov [ebp+12], edi ; 18/02/2015
1638 <1>
1639 0000472A 0C10 <1> or al, MED_DET ; SET MEDIA ESTABLISHED
1640 0000472C 5F <1> pop edi
1641 0000472D 80A7[33770100]0F <1> and byte [DSK_STATE+edi], ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR STATE
1642 00004734 0887[33770100] <1> or [DSK_STATE+edi], al
1643 <1> ; mov ax, cs ; SEGMENT OF MEDIA/DRIVE PARAMETER TABLE
1644 <1> ; mov es, ax ; ES IS SEGMENT OF TABLE
1645 <1> SM_RTN: <1>
1646 <1> ; 06/08/2022
1647 <1> ; call XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
1648 <1> ; call SETUP_END ; VARIOUS CLEANUPS
1649 <1> ; ret
1650 <1> ; 06/08/2022
1651 0000473A E93DFDFFFF <1> jmp SETUP_END
1652 <1>
1653 <1>
1654 <1> ; -----
1655 <1> ; DR_TYPE_CHECK :
1656 <1> ; CHECK IF THE GIVEN DRIVE TYPE IN REGISTER (AL) :
1657 <1> ; IS SUPPORTED IN BIOS DRIVE TYPE TABLE :
1658 <1> ; ON ENTRY: :
1659 <1> ; AL = DRIVE TYPE :
1660 <1> ; ON EXIT: :
1661 <1> ; CY = 0 DRIVE TYPE SUPPORTED :
1662 <1> ; EBX = OFFSET TO MEDIA/DRIVE PARAMETER TABLE :
1663 <1> ; CY = 1 DRIVE TYPE NOT SUPPORTED :
1664 <1> ; REGISTERS ALTERED: EBX, AH ; 11/07/2022 :
1665 <1> ; -----
1666 <1> DR_TYPE_CHECK: <1>
1667 <1> ; 09/08/2022 - TRDOS 386 kernel v2.0.5
1668 <1> ; 12/07/2022
1669 <1> ; 11/07/2022
1670 <1> ; 08/07/2022 - Retro UNIX 386 v1.1 (kernel v0.2.1.5)
1671 <1> ; 24/12/2021
1672 <1> ; push eax ; 11/07/2022
1673 <1> ; push ecx ; 08/07/2022
1674 0000473F BB[F8650000] <1> xor ebx, ebx ; EBX = INDEX TO DR_TYPE TABLE
1675 <1> mov ebx, DR_TYPE
1676 <1> ; mov ecx, DR_CNT ; ECX = LOOP COUNT
1677 00004744 B406 <1> mov cl, DR_CNT
1678 <1> mov ah, DR_CNT ; 11/07/2022
1679 <1> TYPE_CHK: <1>
1680 <1> ; mov ah, [DR_TYPE+ebx] ; GET DRIVE TYPE
1681 <1> ; mov ah, [ebx]
1682 00004746 3A03 <1> cmp al, ah ; DRIVE TYPE MATCH?
1683 00004748 740E <1> cmp al, [ebx] ; 11/07/2022
1684 <1> je short DR_TYPE_VALID ; YES, RETURN WITH CARRY RESET
1685 0000474A 83C305 <1> ; 16/02/2015 (32 bit address modification)
1686 <1> add ebx, 5 ; CHECK NEXT DRIVE TYPE
1687 <1> ; loop TYPE_CHK
1688 0000474D FECC <1> dec cl
1689 0000474F 75F5 <1> dec ah ; 11/07/2022
1690 <1> jnz short TYPE_CHK
1691 00004751 BB[57660000] <1> ;
1692 <1> mov ebx, MD_TBL6 ; 1.44MB fd parameter table
1693 <1> ; Default for GET_PARM (11/12/2014)
1694 00004756 F9 <1> ;
1695 <1> stc ; DRIVE TYPE NOT FOUND IN TABLE
1696 <1> ; jmp short TYPE_RTN
1697 00004757 C3 <1> ; 12/07/2022
1698 <1> ret
1699 <1> DR_TYPE_VALID: <1>
1700 00004758 43 <1> ; mov ebx, [DR_TYPE+ebx+1] ; EBX = MEDIA TABLE
1701 00004759 8B1B <1> inc ebx
1702 <1> mov ebx, [ebx]
1703 <1> TYPE_RTN: <1>
1704 <1> ; pop ecx ; 08/07/2022
1705 <1> ; 24/12/2021
1706 0000475B C3 <1> ; pop eax ; 11/07/2022
1707 <1> ret
1708 <1>
1709 <1> ; -----
1710 <1> ; SEND_SPEC :
1711 <1> ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM :
1712 <1> ; THE DRIVE PARAMETER TABLE POINTED BY @DISK_POINTER :
1713 <1> ; ON ENTRY: @DISK_POINTER = DRIVE PARAMETER TABLE :
1714 <1> ; ON EXIT: NONE :
1715 <1> ; REGISTERS ALTERED: CX, DX :
1716 <1> ; -----
1717 <1> SEND_SPEC: <1>
1718 0000475C 50 <1> ; 06/08/2022
1719 0000475D B8[83470000] <1> push eax ; SAVE AX
1720 00004762 50 <1> mov eax, SPECBAC ; LOAD ERROR ADDRESS
1721 00004763 B403 <1> push eax ; PUSH NEC_OUT ERROR RETURN
1722 00004765 E898050000 <1> mov ah, 03h ; SPECIFY COMMAND
1723 <1> call NEC_OUTPUT ; OUTPUT THE COMMAND
1724 <1> ; sub dl, dl ; FIRST SPECIFY BYTE
1725 0000476A 28C0 <1> ; 06/08/2022
<1> sub al, al ; 0

```



```

1726 0000476C E89B040000 <1> call GET_PARM ; GET PARAMETER TO AH
1727 00004771 E88C050000 <1> call NEC_OUTPUT ; OUTPUT THE COMMAND
1728 <1> ;mov dl, 1 ; SECOND SPECIFY BYTE
1729 <1> ; 06/08/2022
1730 00004776 B001 <1> mov al, 1
1731 00004778 E88F040000 <1> call GET_PARM ; GET PARAMETER TO AH
1732 0000477D E880050000 <1> call NEC_OUTPUT ; OUTPUT THE COMMAND
1733 00004782 58 <1> pop eax ; POP ERROR RETURN
1734 <1> SPECBAC:
1735 00004783 58 <1> pop eax ; RESTORE ORIGINAL AX VALUE
1736 00004784 C3 <1> retn
1737 <1>
1738 <1> ;-----
1739 <1> ; SEND_SPEC_MD :
1740 <1> ; SEND THE SPECIFY COMMAND TO CONTROLLER USING DATA FROM :
1741 <1> ; THE MEDIA/DRIVE PARAMETER TABLE POINTED BY (CS:BX) :
1742 <1> ; ON ENTRY: CS:BX = MEDIA/DRIVE PARAMETER TABLE :
1743 <1> ; ON EXIT: NONE :
1744 <1> ; REGISTERS ALTERED: AX :
1745 <1> ;-----
1746 <1> SEND_SPEC_MD:
1747 00004785 50 <1> push eax ; SAVE RATE DATA
1748 00004786 B8[A3470000] <1> mov eax, SPEC_ESBAC ; LOAD ERROR ADDRESS
1749 0000478B 50 <1> push eax ; PUSH NEC_OUT ERROR RETURN
1750 0000478C B403 <1> mov ah, 03h ; SPECIFY COMMAND
1751 0000478E E86F050000 <1> call NEC_OUTPUT ; OUTPUT THE COMMAND
1752 00004793 8A23 <1> mov ah, [ebx+MD.SPEC1] ; GET 1ST SPECIFY BYTE
1753 00004795 E868050000 <1> call NEC_OUTPUT ; OUTPUT THE COMMAND
1754 0000479A 8A6301 <1> mov ah, [ebx+MD.SPEC2] ; GET SECOND SPECIFY BYTE
1755 0000479D E860050000 <1> call NEC_OUTPUT ; OUTPUT THE COMMAND
1756 000047A2 58 <1> pop eax ; POP ERROR RETURN
1757 <1> SPEC_ESBAC:
1758 000047A3 58 <1> pop eax ; RESTORE ORIGINAL AX VALUE
1759 000047A4 C3 <1> retn
1760 <1>
1761 <1> ;-----
1762 <1> ; XLAT_NEW
1763 <1> ; TRANSLATES DISKETTE STATE LOCATIONS FROM COMPATIBLE
1764 <1> ; MODE TO NEW ARCHITECTURE.
1765 <1>
1766 <1> ; ON ENTRY: DI = DRIVE #
1767 <1> ;-----
1768 <1> XLAT_NEW:
1769 <1> ; 06/08/2022 - TRDOS 386 kernel v2.0.5
1770 <1> ; 11/07/2022 - Retro UNIX 386 v1.1 (kernel v0.2.1.5)
1771 000047A5 83FF01 <1> cmp edi, 1 ; VALID DRIVE
1772 000047A8 7709 <1> ja short XN_OUT ; IF INVALID BACK
1773 000047AA 80BF[33770100]00 <1> cmp byte [DSK_STATE+edi], 0 ; NO DRIVE ?
1774 000047B1 7401 <1> jz short DO_DET ; IF NO DRIVE ATTEMPT DETERMINE
1775 <1>
1776 <1> ; ;mov cx, di ; CX = DRIVE NUMBER
1777 <1> ; ;mov ecx, edi
1778 <1> ; ;or cl, cl
1779 <1> ; ;jz short XN_0
1780 <1> ; ;shl cl, 2 ; CL = SHIFT COUNT, A=0, B=4
1781 <1> ; ;mov al, [HF_CNTRL] ; DRIVE INFORMATION
1782 <1> ; ;ror al, cl ; TO LOW NIBBLE
1783 <1> ;XN_0:
1784 <1> ; ;and al, DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1785 <1> ; ;and byte [DSK_STATE+edi], ~(DRV_DET+FMT_CAPA+TRK_CAPA)
1786 <1> ; ;or [DSK_STATE+edi], al ; UPDATE DRIVE STATE
1787 <1> XN_OUT:
1788 000047B3 C3 <1> retn
1789 <1>
1790 <1> DO_DET:
1791 <1> ;call DRIVE_DET ; TRY TO DETERMINE
1792 <1> ;retn
1793 <1> ;jmp DRIVE_DET
1794 <1>
1795 <1> ;-----
1796 <1> ; DRIVE_DET
1797 <1> ; DETERMINES WHETHER DRIVE IS 80 OR 40 TRACKS AND
1798 <1> ; UPDATES STATE INFORMATION ACCORDINGLY.
1799 <1> ; ON ENTRY: DI = DRIVE #
1800 <1> ;-----
1801 <1> DRIVE_DET:
1802 <1> ; 06/08/2022 - TRDOS 386 kernel v2.0.5
1803 <1> ; 08/07/2022 - Retro UNIX 386 v1.1 (kernel v0.2.1.5)
1804 000047B4 E88E040000 <1> call MOTOR_ON ; TURN ON MOTOR IF NOT ALREADY ON
1805 000047B9 E8FA050000 <1> call RECAL ; RECALIBRATE DRIVE
1806 000047BE 724E <1> jc short DD_BAC ; ASSUME NO DRIVE PRESENT
1807 000047C0 B530 <1> mov ch, TRK_SLAP ; SEEK TO TRACK 48
1808 000047C2 E872050000 <1> call SEEK
1809 000047C7 7245 <1> jc short DD_BAC ; ERROR NO DRIVE
1810 000047C9 B50B <1> mov ch, QUIET_SEEK+1 ; SEEK TO TRACK 10
1811 <1> SK_GIN:
1812 000047CB FECD <1> dec ch ; DECREMENT TO NEXT TRACK
1813 <1> ;push cx ; SAVE TRACK
1814 <1> ; 11/04/2021
1815 000047CD 51 <1> push ecx
1816 000047CE E866050000 <1> call SEEK
1817 000047D3 723A <1> jc short POP_BAC ; POP AND RETURN
1818 000047D5 B8[0F480000] <1> mov eax, POP_BAC ; LOAD NEC OUTPUT ERROR ADDRESS
1819 000047DA 50 <1> push eax
1820 000047DB B404 <1> mov ah, SENSE_DRV_ST ; SENSE DRIVE STATUS COMMAND BYTE
1821 000047DD E820050000 <1> call NEC_OUTPUT ; OUTPUT TO NEC
1822 <1> ;mov ax, di ; AL = DRIVE
1823 <1> ; 06/08/2022
1824 000047E2 89F8 <1> mov eax, edi
1825 000047E4 88C4 <1> mov ah, al ; AH = DRIVE
1826 000047E6 E817050000 <1> call NEC_OUTPUT ; OUTPUT TO NEC
1827 000047EB E849060000 <1> call RESULTS ; GO GET STATUS
1828 000047F0 58 <1> pop eax ; THROW AWAY ERROR ADDRESS
1829 <1> ;pop cx ; RESTORE TRACK
1830 <1> ; 11/04/2021
1831 000047F1 59 <1> pop ecx
1832 000047F2 F605[29770100]10 <1> test byte [NEC_STATUS], HOME ; TRACK 0 ?
1833 000047F9 74D0 <1> jz short SK_GIN ; GO TILL TRACK 0
1834 000047FB 08ED <1> or ch, ch ; IS HOME AT TRACK 0
1835 000047FD 7408 <1> jz short IS_80 ; MUST BE 80 TRACK DRIVE
1836 <1>
1837 <1> ; DRIVE IS A 360; SET DRIVE TO DETERMINED;
1838 <1> ; SET MEDIA TO DETERMINED AT RATE 250.
1839 <1>
1840 000047FF 808F[33770100]94 <1> or byte [DSK_STATE+edi], DRV_DET+MED_DET+RATE_250
1841 00004806 C3 <1> retn ; ALL INFORMATION SET
1842 <1> IS_80:
1843 00004807 808F[33770100]01 <1> or byte [DSK_STATE+edi], TRK_CAPA ; SETUP 80 TRACK CAPABILITY
1844 <1> DD_BAC:
1845 0000480E C3 <1> retn
1846 <1> POP_BAC:
1847 <1> ;pop cx ; THROW AWAY
1848 <1> ; 11/04/2021
1849 0000480F 59 <1> pop ecx

```

```

1850 00004810 c3      <1>      retn
1851                  <1>
1852                  <1>
1853                  <1>      ; 06/08/2022 - TRDOS 386 kernel v2.0.5
1854                  <1>
1855                  <1>      %if 0
1856                  <1>
1857                  <1>      ;-----
1858                  <1>      ; XLAT_OLD
1859                  <1>      ; TRANSLATES DISKETTE STATE LOCATIONS FROM NEW
1860                  <1>      ; ARCHITECTURE TO COMPATIBLE MODE.
1861                  <1>
1862                  <1>      ; ON ENTRY: DI = DRIVE
1863                  <1>      ;-----
1864                  <1>      XLAT_OLD:
1865                  <1>      cmp     edi, 1          ; VALID DRIVE ?
1866                  <1>      ja      short XO_OUT      ; IF INVALID BACK
1867                  <1>      cmp     byte [DSK_STATE+edi], 0      ; NO DRIVE ?
1868                  <1>      jz      short XO_OUT      ; IF NO DRIVE TRANSLATE DONE
1869                  <1>
1870                  <1>      ;-----      TEST FOR SAVED DRIVE INFORMATION ALREADY SET
1871                  <1>
1872                  <1>      ;mov     cx, di          ; CX = DRIVE NUMBER
1873                  <1>      ; 06/08/2022
1874                  <1>      mov     ecx, edi
1875                  <1>      shl     cl, 2          ; CL = SHIFT COUNT, A=0, B=4
1876                  <1>      mov     ah, FMT_CAPA      ; LOAD MULTIPLE DATA RATE BIT MASK
1877                  <1>      ror     ah, cl          ; ROTATE BY MASK
1878                  <1>      test    [HF_CNTRL], ah      ; MULTIPLE-DATA RATE DETERMINED ?
1879                  <1>      jnz     short SAVE_SET      ; IF SO, NO NEED TO RE-SAVE
1880                  <1>
1881                  <1>      ;-----      ERASE DRIVE BITS IN @HF_CNTRL FOR THIS DRIVE
1882                  <1>
1883                  <1>      mov     ah, DRV_DET+FMT_CAPA+TRK_CAPA ; MASK TO KEEP
1884                  <1>      ror     ah, cl          ; FIX MASK TO KEEP
1885                  <1>      not     ah          ; TRANSLATE MASK
1886                  <1>      and     [HF_CNTRL], ah      ; KEEP BITS FROM OTHER DRIVE INTACT
1887                  <1>
1888                  <1>      ;-----      ACCESS CURRENT DRIVE BITS AND STORE IN @HF_CNTRL
1889                  <1>
1890                  <1>      mov     al, [DSK_STATE+edi] ; ACCESS STATE
1891                  <1>      and     al, DRV_DET+FMT_CAPA+TRK_CAPA ; KEEP DRIVE BITS
1892                  <1>      ror     al, cl          ; FIX FOR THIS DRIVE
1893                  <1>      or      [HF_CNTRL], al      ; UPDATE SAVED DRIVE STATE
1894                  <1>
1895                  <1>      ;-----      TRANSLATE TO COMPATIBILITY MODE
1896                  <1>
1897                  <1>      SAVE_SET:
1898                  <1>      mov     ah, [DSK_STATE+edi] ; ACCESS STATE
1899                  <1>      mov     bh, ah          ; TO BH FOR LATER
1900                  <1>      and     ah, RATE_MSK      ; KEEP ONLY RATE
1901                  <1>      cmp     ah, RATE_500      ; RATE 500 ?
1902                  <1>      jz      short CHK_144      ; YES 1.2/1.2 OR 1.44/1.44
1903                  <1>      mov     al, M3D1U      ; AL = 360 IN 1.2 UNESTABLISHED
1904                  <1>      cmp     ah, RATE_300      ; RATE 300 ?
1905                  <1>      jnz     short CHK_250      ; NO, 360/360, 720/720 OR 720/1.44
1906                  <1>      test    bh, DBL_STEP      ; CHECK FOR DOUBLE STEP
1907                  <1>      jnz     short TST_DET      ; MUST BE 360 IN 1.2
1908                  <1>      UNKNO:
1909                  <1>      mov     al, MED_UNK      ; NONE OF THE ABOVE
1910                  <1>      jmp     short AL_SET      ; PROCESS COMPLETE
1911                  <1>      CHK_144:
1912                  <1>      call    CMOS_TYPE      ; RETURN DRIVE TYPE IN (AL)
1913                  <1>      ;;20/02/2015
1914                  <1>      ;;jc     short UNKNO      ; ERROR, SET 'NONE OF ABOVE'
1915                  <1>      jz      short UNKNO ;; 20/02/2015
1916                  <1>      cmp     al, 2          ; 1.2MB DRIVE ?
1917                  <1>      jne     short UNKNO      ; NO, GO SET 'NONE OF ABOVE'
1918                  <1>      mov     al, M1D1U      ; AL = 1.2 IN 1.2 UNESTABLISHED
1919                  <1>      jmp     short TST_DET
1920                  <1>      CHK_250:
1921                  <1>      mov     al, M3D3U      ; AL = 360 IN 360 UNESTABLISHED
1922                  <1>      cmp     ah, RATE_250      ; RATE 250 ?
1923                  <1>      jnz     short UNKNO      ; IF SO FALL IHRU
1924                  <1>      test    bh, TRK_CAPA      ; 80 TRACK CAPABILITY ?
1925                  <1>      jnz     short UNKNO      ; IF SO JUMP, FALL THRU TEST DET
1926                  <1>      TST_DET:
1927                  <1>      test    bh, MED_DET      ; DETERMINED ?
1928                  <1>      jz      short AL_SET      ; IF NOT THEN SET
1929                  <1>      add     al, 3          ; MAKE DETERMINED/ESTABLISHED
1930                  <1>      AL_SET:
1931                  <1>      and     byte [DSK_STATE+edi], ~(DRV_DET+FMT_CAPA+TRK_CAPA) ; CLEAR DRIVE
1932                  <1>      or      [DSK_STATE+edi], al      ; REPLACE WITH COMPATIBLE MODE
1933                  <1>      XO_OUT:
1934                  <1>      retn
1935                  <1>
1936                  <1>      %endif
1937                  <1>
1938                  <1>      ;-----
1939                  <1>      ; SETUP_STATE:      INITIALIZES START AND END RATES.
1940                  <1>      ;-----
1941                  <1>      SETUP_STATE:
1942                  <1>      test    byte [DSK_STATE+edi], MED_DET ; MEDIA DETERMINED ?
1943                  <1>      jnz     short J1C      ; NO STATES IF DETERMINED
1944                  <1>      mov     ax, (RATE_500*256)+RATE_300 ; AH = START RATE, AL = END RATE
1945                  <1>      test    byte [DSK_STATE+edi], DRV_DET ; DRIVE ?
1946                  <1>      jz      short AX_SET      ; DO NOT KNOW DRIVE
1947                  <1>      test    byte [DSK_STATE+edi], FMT_CAPA ; MULTI-RATE?
1948                  <1>      jnz     short AX_SET      ; JUMP IF YES
1949                  <1>      mov     ax, RATE_250*257 ; START A END RATE 250 FOR 360 DRIVE
1950                  <1>      AX_SET:
1951                  <1>      and     byte [DSK_STATE+edi], ~(RATE_MSK+DBL_STEP) ; TURN OFF THE RATE
1952                  <1>      or      [DSK_STATE+edi], ah      ; RATE FIRST TO TRY
1953                  <1>      and     byte [LASTRATE], ~STRT_MSK ; ERASE LAST TO TRY RATE BITS
1954                  <1>      ror     al, 4          ; TO OPERATION LAST RATE LOCATION
1955                  <1>      or      [LASTRATE], al      ; LAST RATE
1956                  <1>      J1C:
1957                  <1>      retn
1958                  <1>
1959                  <1>      ;-----
1960                  <1>      ; FMT_INIT: ESTABLISH STATE IF UNESTABLISHED AT FORMAT TIME.
1961                  <1>      ;-----
1962                  <1>      FMT_INIT:
1963                  <1>      test    byte [DSK_STATE+edi], MED_DET ; IS MEDIA ESTABLISHED
1964                  <1>      jnz     short F1_OUT      ; IF SO RETURN
1965                  <1>      call    CMOS_TYPE      ; RETURN DRIVE TYPE IN AL
1966                  <1>      ;; 20/02/2015
1967                  <1>      ;;jc     short CL_DRV      ; ERROR IN CMOS ASSUME NO DRIVE
1968                  <1>      jz      short CL_DRV ;; 20/02/2015
1969                  <1>      dec     al          ; MAKE ZERO ORIGIN
1970                  <1>      ;;JS     short CL_DRV      ; NO DRIVE IF AL 0
1971                  <1>      mov     ah, [DSK_STATE+edi] ; AH = CURRENT STATE
1972                  <1>      and     ah, ~(MED_DET+DBL_STEP+RATE_MSK) ; CLEAR
1973                  <1>      or      al, al          ; CHECK FOR 360

```

```

1974 0000486F 7505      <1>      jnz      short N_360          ; IF 360 WILL BE 0
1975 00004871 80CC90    <1>      or       ah, MED_DET+RATE_250 ; ESTABLISH MEDIA
1976 00004874 EB25      <1>      jmp      short SKP_STATE          ; SKIP OTHER STATE PROCESSING
1977                                <1> N_360:
1978 00004876 FEC8      <1>      dec      al              ; 1.2 M DRIVE
1979 00004878 7505      <1>      jnz      short N_12          ; JUMP IF NOT
1980                                <1> F1_RATE:
1981 0000487A 80CC10    <1>      or       ah, MED_DET+RATE_500 ; SET FORMAT RATE
1982 0000487D EB1C      <1>      jmp      short SKP_STATE          ; SKIP OTHER STATE PROCESSING
1983                                <1> N_12:
1984 0000487F FEC8      <1>      dec      al              ; CHECK FOR TYPE 3
1985 00004881 750F      <1>      jnz      short N_720          ; JUMP IF NOT
1986 00004883 F6C404    <1>      test     ah, DRV_DET          ; IS DRIVE DETERMINED
1987 00004886 7410      <1>      jz       short ISNT_12         ; TREAT AS NON 1.2 DRIVE
1988 00004888 F6C402    <1>      test     ah, FMT_CAPA         ; IS 1.2M
1989 0000488B 740B      <1>      jz       short ISNT_12         ; JUMP IF NOT
1990 0000488D 80CC50    <1>      or       ah, MED_DET+RATE_300 ; RATE 300
1991 00004890 EB09      <1>      jmp      short SKP_STATE          ; CONTINUE
1992                                <1> N_720:
1993 00004892 FEC8      <1>      dec      al              ; CHECK FOR TYPE 4
1994 00004894 750C      <1>      jnz      short CL_DRV          ; NO DRIVE, CMOS BAD
1995 00004896 EBE2      <1>      jmp      short F1_RATE
1996                                <1> ISNT_12:
1997 00004898 80CC90    <1>      or       ah, MED_DET+RATE_250 ; MUST BE RATE 250
1998                                <1> SKP_STATE:
1999 0000489B 88A7[33770100] <1>      mov      [DSK_STATE+edi], ah ; STORE AWAY
2000                                <1> F1_OUT:
2001 000048A1 C3        <1>      retn
2002                                <1> CL_DRV:
2003 000048A2 30E4      <1>      xor      ah, ah              ; CLEAR STATE
2004 000048A4 EBF5      <1>      jmp      short SKP_STATE          ; SAVE IT
2005                                <1>
2006                                <1> ;-----
2007                                <1> ; MED_CHANGE
2008                                <1> ; CHECKS FOR MEDIA CHANGE, RESETS MEDIA CHANGE,
2009                                <1> ; CHECKS MEDIA CHANGE AGAIN.
2010                                <1> ;
2011                                <1> ; ON EXIT: CY = 1 MEANS MEDIA CHANGE OR TIMEOUT
2012                                <1> ; @DSKETTE_STATUS = ERROR CODE
2013                                <1> ;-----
2014                                <1> MED_CHANGE:
2015                                <1> ; 06/08/2022 - TRDOS 386 v2.0.5
2016 000048A6 E8E8050000 <1>      call     READ_DSKCHNG          ; READ DISK CHANCE LINE STATE
2017 000048AB 7445      <1>      jz       short MC_OUT          ; BYPASS HANDLING DISK CHANGE LINE
2018 000048AD 80A7[33770100]EF <1>      and      byte [DSK_STATE+edi], ~MED_DET ; CLEAR STATE FOR THIS DRIVE
2019                                <1>
2020                                <1> ; THIS SEQUENCE ENSURES WHENEVER A DISKETTE IS CHANGED THAT
2021                                <1> ; ON THE NEXT OPERATION THE REQUIRED MOTOR START UP TIME WILL
2022                                <1> ; BE WAITED. (DRIVE MOTOR MAY GO OFF UPON DOOR OPENING).
2023                                <1> ;
2024                                <1> ;mov      cx, di              ; CL = DRIVE 0
2025                                <1> ; 06/08/2022
2026 000048B4 89F9      <1>      mov      ecx, edi
2027 000048B6 B001      <1>      mov      al, 1
2028 000048B8 D2E0      <1>      shl      al, cl              ; MOTOR ON BIT MASK
2029 000048BA F6D0      <1>      not      al              ; TO APPROPRIATE POSITION
2030 000048BC FA        <1>      cli              ; KEEP ALL BUT MOTOR ON
2031 000048BD 2005[26770100] <1>      and      [MOTOR_STATUS], al ; NO INTERRUPTS
2032 000048C3 FB        <1>      sti              ; TURN MOTOR OFF INDICATOR
2033 000048C4 E87E030000 <1>      call     MOTOR_ON              ; INTERRUPTS ENABLED
2034                                <1> ; TURN MOTOR ON
2035                                <1> ;----- THIS SEQUENCE OF SEEKS IS USED TO RESET DISKETTE CHANGE SIGNAL
2036                                <1> ;
2037 000048C9 E82FFAFFFF <1>      call     DSK_RESET              ; RESET NEC
2038 000048CE B501      <1>      mov      ch, 1              ; MOVE TO CYLINDER 1
2039 000048D0 E864040000 <1>      call     SEEK                  ; ISSUE SEEK
2040 000048D5 30ED      <1>      xor      ch, ch              ; MOVE TO CYLINDER 0
2041 000048D7 E85D040000 <1>      call     SEEK                  ; ISSUE SEEK
2042 000048DC C605[28770100]06 <1>      mov      byte [DSKETTE_STATUS], MEDIA_CHANGE ; STORE IN STATUS
2043                                <1> OK1:
2044 000048E3 E8AB050000 <1>      call     READ_DSKCHNG          ; CHECK MEDIA CHANGED AGAIN
2045 000048E8 7407      <1>      jz       short OK2              ; IF ACTIVE, NO DISKETTE, TIMEOUT
2046                                <1> OK4:
2047 000048EA C605[28770100]80 <1>      mov      byte [DSKETTE_STATUS], TIME_OUT ; TIMEOUT IF DRIVE EMPTY
2048                                <1> OK2:
2049 000048F1 F9        <1>      stc
2050                                <1> MC_OUT: ; 06/08/2022 (cf = 0)
2051 000048F2 C3        <1>      retn
2052                                <1> ;MC_OUT:
2053                                <1> ;clc
2054                                <1> ;retn
2055                                <1>
2056                                <1> ;-----
2057                                <1> ; SEND_RATE
2058                                <1> ; SENDS DATA RATE COMMAND TO NEC
2059                                <1> ; ON ENTRY: DI = DRIVE #
2060                                <1> ; ON EXIT: NONE
2061                                <1> ; REGISTERS ALTERED: DX
2062                                <1> ;-----
2063                                <1> SEND_RATE:
2064                                <1> ;push     ax
2065                                <1> ; 11/04/2021
2066 000048F3 50        <1>      push     eax
2067 000048F4 8025[30770100]3F <1>      and      byte [LASTRATE], ~SEND_MSK ; ELSE CLEAR LAST RATE ATTEMPTED
2068 000048FB 8A87[33770100] <1>      mov      al, [DSK_STATE+edi] ; GET RATE STATE OF THIS DRIVE
2069 00004901 24C0      <1>      and      al, SEND_MSK          ; KEEP ONLY RATE BITS
2070 00004903 0805[30770100] <1>      or       [LASTRATE], al        ; SAVE NEW RATE FOR NEXT CHECK
2071 00004909 C0C002    <1>      rol      al, 2              ; MOVE TO BIT OUTPUT POSITIONS
2072 0000490C 66BAF703 <1>      mov      dx, 03F7h          ; OUTPUT NEW DATA RATE
2073 00004910 EE        <1>      out      dx, al
2074                                <1> ;pop      ax
2075                                <1> ; 11/04/2021
2076 00004911 58        <1>      pop      eax
2077 00004912 C3        <1>      retn
2078                                <1>
2079                                <1> ;-----
2080                                <1> ; CHK_LASTRATE
2081                                <1> ; CHECK PREVIOUS DATE RATE SNT TO THE CONTROLLER.
2082                                <1> ; ON ENTRY:
2083                                <1> ; DI = DRIVE #
2084                                <1> ; ON EXIT:
2085                                <1> ; ZF = 1 DATA RATE IS THE SAME AS THE LAST RATE SENT TO NEC
2086                                <1> ; ZF = 0 DATA RATE IS DIFFERENT FROM LAST RATE
2087                                <1> ; REGISTERS ALTERED: DX
2088                                <1> ;-----
2089                                <1> CHK_LASTRATE:
2090                                <1> ; 13/07/2022 - TRDOS 386 v2.0.5
2091                                <1> ;push     ax
2092                                <1> ; 11/04/2021
2093 00004913 50        <1>      push     eax
2094                                <1> ; 13/07/2022 (BugFix)
2095 00004914 8A25[30770100] <1>      mov      ah, [LASTRATE]          ; GET LAST DATA RATE SELECTED
2096 0000491A 8A87[33770100] <1>      mov      al, [DSK_STATE+edi] ; GET RATE STATE OF THIS DRIVE
2097 00004920 6625C0C0 <1>      and      ax, SEND_MSK*257      ; KEEP ONLY RATE BITS OF BOTH

```

```

2098 00004924 38E0      <1>      cmp     al, ah          ; COMPARE TO PREVIOUSLY TRIED
2099                  <1>                  ; ZF = 1 RATE IS THE SAME
2100                  <1>      ;pop     ax          ; RESTORE REG.
2101                  <1>      ; 11/04/2021
2102 00004926 58        <1>      pop     eax
2103 00004927 C3        <1>      retn
2104                  <1>
2105                  <1>-----
2106                  <1>      DMA_SETUP
2107                  <1>      THIS ROUTINE SETS UP THE DMA FOR READ/WRITE/VERIFY OPERATIONS.
2108                  <1>
2109                  <1>      ON ENTRY: AL = DMA COMMAND
2110                  <1>
2111                  <1>      ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
2112                  <1>-----
2113                  <1>
2114                  <1>      ; SI = Head #, # of Sectors or DASD Type
2115                  <1>
2116                  <1>      ; 22/08/2015
2117                  <1>      ; 08/02/2015 - Protected Mode Modification
2118                  <1>      ; 06/02/2015 - 07/02/2015
2119                  <1>      ; NOTE: Buffer address must be in 1st 16MB of Physical Memory (24 bit limit).
2120                  <1>      ; (DMA Address = Physical Address)
2121                  <1>      ; (Retro UNIX 386 v1 kernel/System Mode Virtual Address = Physical Address)
2122                  <1>
2123                  <1>      ; 09/08/2022
2124                  <1>      ; 06/08/2022
2125                  <1>      ; 04/02/2016 (c1c)
2126                  <1>      ; 20/02/2015 modification (source: AWARD BIOS 1999, DMA_SETUP)
2127                  <1>      ; 16/12/2014 (IODELAY)
2128                  <1>
2129                  <1>      DMA_SETUP:
2130                  <1>      ; 09/08/2022
2131                  <1>      ; 06/08/2022 - TRDOS 386 kernel v2.0.5
2132                  <1>      ; 20/02/2015
2133                  <1>      ; mov     edx, [ebp+4]          ; Buffer address
2134                  <1>      ; 06/08/2022
2135                  <1>      ; mov     edx, ebp ; buffer address
2136                  <1>      ; test    edx, 0FF00000h          ; 16 MB limit (22/08/2015, bugfix)
2137 00004928 F7C500000FF <1>      test    ebp, 0FF00000h
2138 0000492E 7566       <1>      jnz     short dma_bnd_err_stc
2139                  <1>      ; 09/08/2022
2140 00004930 89EA       <1>      mov     edx, ebp
2141                  <1>
2142                  <1>      ; push    ax          ; DMA command
2143                  <1>      ; 11/04/2021
2144 00004932 50        <1>      push    eax
2145                  <1>      ; 06/08/2022
2146                  <1>      ; push    edx          ; *
2147                  <1>      ; mov     dl, 3          ; GET BYTES/SECTOR PARAMETER
2148                  <1>      ; 06/08/2022
2149 00004933 B003       <1>      mov     al, 3          ; GET BYTES/SECTOR PARAMETER
2150 00004935 E8D2020000 <1>      call    GET_PARM
2151 0000493A 88E1       <1>      mov     cl, ah          ; SHIFT COUNT (0=128, 1=256, 2=512 ETC)
2152                  <1>      ; mov     ax, si          ; Sector count
2153                  <1>      ; 06/08/2022
2154 0000493C 89F0       <1>      mov     eax, esi
2155 0000493E 88C4       <1>      mov     ah, al
2156 00004940 28C0       <1>      sub     al, al          ; AH = # OF SECTORS
2157                  <1>      ; shr     ax, 1          ; AL = 0, AX = # SECTORS * 256
2158                  <1>      ; 06/08/2022          ; AX = # SECTORS * 128
2159 00004942 D1E8       <1>      shr     eax, 1
2160                  <1>      ; shl     ax, cl          ; SHIFT BY PARAMETER VALUE
2161                  <1>      ; dec     ax          ; -1 FOR DMA VALUE
2162                  <1>      ; mov     cx, ax
2163 00004944 D3E0       <1>      shl     eax, cl
2164 00004946 48        <1>      dec     eax
2165 00004947 89C1       <1>      mov     ecx, eax
2166                  <1>      ; 06/08/2022
2167                  <1>      ; pop     edx          ; *
2168                  <1>      ; pop     ax
2169                  <1>      ; 11/04/2021
2170 00004949 58        <1>      pop     eax
2171 0000494A 3C42       <1>      cmp     al, 42h
2172 0000494C 7507       <1>      jne     short NOT_VERF
2173                  <1>      ; 09/08/2022
2174 0000494E BA0000FF00 <1>      mov     edx, 0FF0000h
2175                  <1>      ; 06/08/2022
2176                  <1>      ; mov     ebp, 0FF0000h
2177 00004953 EB08       <1>      jmp     short J33
2178                  <1>      NOT_VERF:
2179 00004955 6601CA     <1>      add     dx, cx          ; check for overflow
2180 00004958 723D       <1>      jc     short dma_bnd_err
2181                  <1>
2182 0000495A 6629CA     <1>      sub     dx, cx          ; Restore start address
2183                  <1>      J33:
2184 0000495D FA        <1>      cli          ; DISABLE INTERRUPTS DURING DMA SET-UP
2185 0000495E E60C       <1>      out     DMA+12, al      ; SET THE FIRST/LAST F/F
2186                  <1>      IODELAY          ; WAIT FOR I/O
2187 00004960 EB00       <2>      jmp     short $+2
2188 00004962 EB00       <2>      jmp     short $+2
2189                  <1>      out     DMA+11, al      ; OUTPUT THE MODE BYTE
2190                  <1>      ; mov     eax, edx          ; Buffer address
2191                  <1>      ; 06/08/2022
2192 00004966 89D0       <1>      mov     eax, ebp
2193 00004968 E604       <1>      mov     eax, edx
2194                  <1>      out     DMA+4, al      ; OUTPUT LOW ADDRESS
2195                  <1>      IODELAY          ; WAIT FOR I/O
2196 0000496A EB00       <2>      jmp     short $+2
2197 0000496C EB00       <2>      jmp     short $+2
2198 0000496E 88E0       <1>      mov     al, ah
2199 00004970 E604       <1>      out     DMA+4, al      ; OUTPUT HIGH ADDRESS
2200 00004972 C1E810     <1>      shr     eax, 16
2201                  <1>      IODELAY          ; I/O WAIT STATE
2202 00004975 EB00       <2>      jmp     short $+2
2203 00004977 EB00       <2>      jmp     short $+2
2204 00004979 E681       <1>      out     081h, al      ; OUTPUT HIGHEST BITS TO PAGE REGISTER
2205                  <1>      IODELAY
2206 0000497B EB00       <2>      jmp     short $+2
2207 0000497D EB00       <2>      jmp     short $+2
2208                  <1>      ; mov     ax, cx          ; Byte count - 1
2209                  <1>      ; 06/08/2022
2210 0000497F 89C8       <1>      mov     eax, ecx
2211 00004981 E605       <1>      out     DMA+5, al      ; LOW BYTE OF COUNT
2212                  <1>      IODELAY          ; WAIT FOR I/O
2213 00004983 EB00       <2>      jmp     short $+2
2214 00004985 EB00       <2>      jmp     short $+2
2215 00004987 88E0       <1>      mov     al, ah
2216 00004989 E605       <1>      out     DMA+5, al      ; HIGH BYTE OF COUNT
2217                  <1>      IODELAY
2218 0000498B EB00       <2>      jmp     short $+2
2219 0000498D EB00       <2>      jmp     short $+2
2220 0000498F FB        <1>      sti          ; RE-ENABLE INTERRUPTS

```

```

2210 00004990 B002      <1>      mov     al, 2          ; MODE FOR 8237
2211 00004992 E60A      <1>      out      DMA+10, al        ; INITIALIZE THE DISKETTE CHANNEL
2212                      <1>
2213 00004994 F8         <1>      cllc     ; 04/02/2016
2214 00004995 C3         <1>      retn
2215                      <1>
2216                      <1> dma_bnd_err_stc:
2217 00004996 F9         <1>      stc
2218                      <1> dma_bnd_err:
2219 00004997 C605[28770100]09 <1>      mov     byte [DSKETTE_STATUS], DMA_BOUNDARY ; SET ERROR
2220 0000499E C3         <1>      retn                ; CY SET BY ABOVE IF ERROR
2221                      <1>
2222                      <1> ; 16/12/2014
2223                      <1> ; CLI
2224                      <1> ; OUT DMA+12,AL
2225                      <1> ; jmp $+2
2226                      <1> ; IODELAY
2227                      <1> ; OUT DMA+11,AL
2228                      <1> ; SIODELAY
2229                      <1> ; cmp AL,42H
2230                      <1> ; jNE short NOT_VERF
2231                      <1> ; xor AX,AX
2232                      <1> ; jmp SHORT J33
2233                      <1> ; NOT_VERF:
2234                      <1> ; mov AX,ES
2235                      <1> ; ROL AX,4
2236                      <1> ; mov CH,AL
2237                      <1> ; and AL,11110000B
2238                      <1> ; add AX,[BP+2]
2239                      <1> ; mov eax,[ebp+4] ; 06/02/2015
2240                      <1> ; jnc short J33
2241                      <1> ; inc CH
2242                      <1> ; J33:
2243                      <1> ; push eax
2244                      <1> ; OUT DMA+4,AL
2245                      <1> ; jmp $+2
2246                      <1> ; IODELAY
2247                      <1> ; mov AL,AH
2248                      <1> ; OUT DMA+4,AL
2249                      <1> ; shr eax, 16 ; 07/02/2015
2250                      <1> ; mov AL,CH
2251                      <1> ; jmp $+2
2252                      <1> ; IODELAY
2253                      <1> ; and AL,00001111B
2254                      <1> ; OUT 081H,AL
2255                      <1> ; SIODELAY
2256                      <1>
2257                      <1> ; ---- DETERMINE COUNT
2258                      <1> ; sub eax, eax ; 08/02/2015
2259                      <1> ; mov AX,SI
2260                      <1> ; xchg AL,AH
2261                      <1> ; sub AL,AL
2262                      <1> ; SHR AX,1
2263                      <1> ; push AX
2264                      <1> ; mov DL,3
2265                      <1> ; call GET_PARM
2266                      <1> ; mov CL,AH
2267                      <1> ; pop AX
2268                      <1> ; SHL AX,CL
2269                      <1> ; dec AX
2270                      <1> ; push eax ; 08/02/2015
2271                      <1> ; OUT DMA+5,AL
2272                      <1> ; jmp $+2
2273                      <1> ; IODELAY
2274                      <1> ; mov AL,AH
2275                      <1> ; OUT DMA+5,AL
2276                      <1> ; SIODELAY
2277                      <1> ; STI
2278                      <1> ; pop ecx ; 08/02/2015
2279                      <1> ; pop eax ; 08/02/2015
2280                      <1> ; add AX, CX
2281                      <1> ; add ecx,eax ; 08/02/2015
2282                      <1> ; mov AL, 2
2283                      <1> ; jmp $+2
2284                      <1> ; SIODELAY
2285                      <1> ; OUT DMA+10,AL
2286                      <1> ; jnc short NO_BAD
2287                      <1> ; jc short dma_bnd_err ; 08/02/2015
2288                      <1> ; and ecx,0FFF0000h ; 16 MB limit
2289                      <1> ; jz short NO_BAD
2290                      <1> ; dma_bnd_err:
2291                      <1> ; mov byte [DSKETTE_STATUS],DMA_BOUNDARY ; SET ERROR
2292                      <1> ; NO_BAD:
2293                      <1> ; retn
2294                      <1>
2295                      <1> ; -----
2296                      <1> ; FMTDMA_SET
2297                      <1> ; THIS ROUTINE SETS UP THE DMA CONTROLLER FOR A FORMAT OPERATION.
2298                      <1>
2299                      <1> ; ON ENTRY: NOTHING REQUIRED
2300                      <1>
2301                      <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
2302                      <1> ; -----
2303                      <1>
2304                      <1> FMTDMA_SET:
2305                      <1> ; 06/08/2022 - TRDOS 386 v2.0.5
2306                      <1> ; 20/02/2015 modification
2307                      <1> ; mov edx, [ebp+4] ; Buffer address
2308                      <1> ; 06/08/2022
2309                      <1> ; mov edx, ebp ; buffer address
2310                      <1> ; 06/08/2022
2311                      <1> ; test edx, 0FF000000h ; 16 MB limit
2312 0000499F F7C5000000FF <1> ; test ebp, 0FF000000h
2313 000049A5 75EF <1> ; jnz short dma_bnd_err_stc
2314                      <1> ;
2315                      <1> ; push dx ; *
2316                      <1> ; 11/04/2021
2317                      <1> ; 06/08/2022
2318                      <1> ; push edx
2319                      <1> ; mov dl, 4
2320                      <1> ; 06/08/2022
2321 000049A7 B004 <1> ; mov al, 4
2322 000049A9 E85E020000 <1> ; call GET_PARM
2323 000049AE 88E0 <1> ; mov al, ah
2324 000049B0 28E4 <1> ; sub ah, ah
2325                      <1> ; shl ax, 2
2326                      <1> ; 06/08/2022
2327 000049B2 C1E002 <1> ; shl eax, 2
2328                      <1> ; dec ax
2329 000049B5 48 <1> ; dec eax
2330                      <1> ; mov cx, ax
2331 000049B6 89C1 <1> ; mov ecx, eax
2332                      <1> ; pop dx ; *
2333                      <1> ; 11/04/2021

```

```

2334      <1>      ; 06/08/2022
2335      <1>      ;pop     edx
2336      <1>      ; 06/08/2022
2337 000049B8 B04A      <1>      mov     al, 04Ah
2338      <1>      ;
2339 000049BA EB99      <1>      jmp     short NOT_VERF ; 06/08/2022
2340      <1>
2341      <1>      ;; 06/08/2022
2342      <1>      add     dx, cx          ; check for overflow
2343      <1>      jc      short dma_bnd_err
2344      <1>      ;
2345      <1>      sub     dx, cx          ; Restore start address
2346      <1>      ;
2347      <1>      ;mov     al, 04Ah          ; WILL WRITE TO THE DISKETTE
2348      <1>      cli          ; DISABLE INTERRUPTS DURING DMA SET-UP
2349      <1>      out     DMA+12, al      ; SET THE FIRST/LA5T F/F
2350      <1>      IODELAY          ; WAIT FOR I/O
2351      <1>      out     DMA+11, al      ; OUTPUT THE MODE BYTE
2352      <1>      mov     eax, edx        ; Buffer address
2353      <1>      out     DMA+4, al      ; OUTPUT LOW ADDRESS
2354      <1>      IODELAY          ; WAIT FOR I/O
2355      <1>      mov     al, ah
2356      <1>      out     DMA+4, al      ; OUTPUT HIGH ADDRESS
2357      <1>      shr     eax, 16
2358      <1>      IODELAY          ; I/O WAIT STATE
2359      <1>      out     081h, al      ; OUTPUT HIGHEST BITS TO PAGE REGISTER
2360      <1>      IODELAY
2361      <1>      ;mov     ax, cx          ; Byte count - 1
2362      <1>      ; 06/08/2022
2363      <1>      mov     eax, ecx
2364      <1>      out     DMA+5, al      ; LOW BYTE OF COUNT
2365      <1>      IODELAY          ; WAIT FOR I/O
2366      <1>      mov     al, ah
2367      <1>      out     DMA+5, al      ; HIGH BYTE OF COUNT
2368      <1>      IODELAY
2369      <1>      sti          ; RE-ENABLE INTERRUPTS
2370      <1>      mov     al, 2          ; MODE FOR 8237
2371      <1>      out     DMA+10, al     ; INITIALIZE THE DISKETTE CHANNEL
2372      <1>
2373      <1>      ; 06/08/2022
2374      <1>      cld
2375      <1>      retn
2376      <1>
2377      <1>      ;; 08/02/2015 - Protected Mode Modification
2378      <1>      mov     AL,04AH          ; WILL WRITE TO THE DISKETTE
2379      <1>      CLI          ; DISABLE INTERRUPTS DURING DMA SET-UP
2380      <1>      OUT     DMA+12,AL        ; SET THE FIRST/LA5T F/F
2381      <1>      ;jmp     $+2          ; WAIT FOR I/O
2382      <1>      IODELAY
2383      <1>      OUT     DMA+11,AL        ; OUTPUT THE MODE BYTE
2384      <1>      ;mov     AX,ES          ; GET THE ES VALUE
2385      <1>      ;ROL     AX,4          ; ROTATE LEFT
2386      <1>      ;mov     CH,AL        ; GET HIGHEST NIBBLE OF ES TO CH
2387      <1>      ;and     AL,11110000B ; ZERO THE LOW NIBBLE FROM SEGMENT
2388      <1>      ;add     AX,[BP+2]     ; TEST FOR CARRY FROM ADDITION
2389      <1>      ;jnc     short J33A
2390      <1>      ;inc     CH          ; CARRY MEANS HIGH 4 BITS MUST BE INC
2391      <1>      ;mov     eax,[ebp+4] ; 08/02/2015
2392      <1>      ;J33A:
2393      <1>      push    eax ; 08/02/2015 ; SAVE START ADDRESS
2394      <1>      OUT     DMA+4,AL        ; OUTPUT LOW ADDRESS
2395      <1>      ;jmp     $+2          ; WAIT FOR I/O
2396      <1>      IODELAY
2397      <1>      mov     AL,AH
2398      <1>      OUT     DMA+4,AL        ; OUTPUT HIGH ADDRESS
2399      <1>      shr     eax,16 ; 08/02/2015
2400      <1>      ;mov     AL,CH        ; GET HIGH 4 BITS
2401      <1>      ;jmp     $+2          ; I/O WAIT STATE
2402      <1>      IODELAY
2403      <1>      ;and     AL,00001111B
2404      <1>      OUT     081H,AL        ; OUTPUT HIGH 4 BITS TO PAGE REGISTER
2405      <1>
2406      <1>      ;----- DETERMINE COUNT
2407      <1>      sub     eax,eax ; 08/02/2015
2408      <1>      mov     DL,4          ; SECTORS/TRACK VALUE IN PARM TABLE
2409      <1>      call    GET_PARM          ; "
2410      <1>      xchg    AL,AH          ; AL = SECTORS/TRACK VALUE
2411      <1>      sub     AH,AH          ; AX = SECTORS/TRACK VALUE
2412      <1>      SHL     AX,2          ; AX = SEC/TRK * 4 (OFFSET C,H,R,N)
2413      <1>      dec     AX          ; -1 FOR DMA VALUE
2414      <1>      push    eax ; 08/02/2015 ; SAVE # OF BYTES TO BE TRANSFERED
2415      <1>      OUT     DMA+5,AL        ; LOW BYTE OF COUNT
2416      <1>      ;jmp     $+2          ; WAIT FOR I/O
2417      <1>      IODELAY
2418      <1>      mov     AL,AH
2419      <1>      OUT     DMA+5,AL        ; HIGH BYTE OF COUNT
2420      <1>      STI          ; RE-ENABLE INTERRUPTS
2421      <1>      pop     ecx ; 08/02/2015 ; RECOVER COUNT VALUE
2422      <1>      pop     eax ; 08/02/2015 ; RECOVER ADDRESS VALUE
2423      <1>      ;add     AX,CX          ; ADD, TEST FOR 64K OVERFLOW
2424      <1>      add     ecx,eax ; 08/02/2015
2425      <1>      mov     AL,2          ; MODE FOR 8237
2426      <1>      ;jmp     $+2          ; WAIT FOR I/O
2427      <1>      IODELAY
2428      <1>      OUT     DMA+10,AL        ; INITIALIZE THE DISKETTE CHANNEL
2429      <1>      ;jnc     short FMTDMA_OK ; CHECK FOR ERROR
2430      <1>      jc      short fmtdma_bnd_err ; 08/02/2015
2431      <1>      and     ecx,0FFF0000h ; 16 MB limit
2432      <1>      jz      short FMTDMA_OK
2433      <1>      stc      ; 20/02/2015
2434      <1>      ;fmtdma_bnd_err:
2435      <1>      mov     byte [DSKETTE_STATUS],DMA_BOUNDARY ; SET ERROR
2436      <1>      ;FMTDMA_OK:
2437      <1>      retn          ; CY SET BY ABOVE IF ERROR
2438      <1>
2439      <1>      ;-----
2440      <1>      NEC_INIT
2441      <1>      THIS ROUTINE SEEKS TO THE REQUESTED TRACK AND INITIALIZES
2442      <1>      THE NEC FOR THE READ/WRITE/VERIFY/FORMAT OPERATION.
2443      <1>
2444      <1>      ON ENTRY: AH = NEC COMMAND TO BE PERFORMED
2445      <1>
2446      <1>      ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
2447      <1>      ;-----
2448      <1>      NEC_INIT:
2449      <1>      ; 11/08/2022
2450      <1>      ; EBX = user's ECX register content (on stack)
2451      <1>      ; 10/08/2022
2452      <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
2453      <1>      push    ax          ; SAVE NEC COMMAND
2454      <1>      ; 11/04/2021
2455      <1>      push    eax
2456      <1>      call    MOTOR_ON          ; TURN MOTOR ON FOR SPECIFIC DRIVE
2457      <1>

```

```

2458 <1> ;----- DO THE SEEK OPERATION
2459 <1>
2460 <1> ;mov ch, [ebp+1] ; CH = TRACK #
2461 <1> ; 10/08/2022
2462 000049C2 88FD <1> mov ch, bh ; CH = TRACK #
2463 000049C4 E870030000 <1> call SEEK ; MOVE TO CORRECT TRACK
2464 <1> ;pop ax ; RECOVER COMMAND
2465 <1> ; 11/04/2021
2466 000049C9 58 <1> pop eax
2467 000049CA 721D <1> jc short ER_1 ; ERROR ON SEEK
2468 000049CC BB[E9490000] <1> mov ebx, ER_1 ; LOAD ERROR ADDRESS
2469 000049D1 53 <1> push ebx ; PUSH NEC_OUT ERROR RETURN
2470 <1>
2471 <1> ;----- SEND OUT THE PARAMETERS TO THE CONTROLLER
2472 <1>
2473 000049D2 E82B030000 <1> call NEC_OUTPUT ; OUTPUT THE OPERATION COMMAND
2474 <1> ;mov ax, si ; AH = HEAD #
2475 <1> ; 06/08/2022
2476 000049D7 89F0 <1> mov eax, esi
2477 000049D9 89FB <1> mov ebx, edi ; BL = DRIVE #
2478 000049DB C0E402 <1> sal ah, 2 ; MOVE IT TO BIT 2
2479 000049DE 80E404 <1> and ah, 00000100b ; ISOLATE THAT BIT
2480 000049E1 08DC <1> or ah, bl ; OR IN THE DRIVE NUMBER
2481 000049E3 E81A030000 <1> call NEC_OUTPUT ; FALL THRU CY SET IF ERROR
2482 000049E8 5B <1> pop ebx ; THROW AWAY ERROR RETURN
2483 <1> ER_1:
2484 000049E9 C3 <1> ret
2485 <1>
2486 <1> ;-----
2487 <1> ; RWV_COM
2488 <1> ; THIS ROUTINE SENDS PARAMETERS TO THE NEC SPECIFIC TO THE
2489 <1> ; READ/WRITE/VERIFY OPERATIONS.
2490 <1>
2491 <1> ; ON ENTRY: CS:BX = ADDRESS OF MEDIA/DRIVE PARAMETER TABLE
2492 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
2493 <1> ;-----
2494 <1> RWV_COM:
2495 <1> ; 11/08/2022
2496 <1> ; 10/08/2022
2497 <1> ; 06/08/2022 - TRDOS 386 v2.0.5
2498 000049EA B8[364A0000] <1> mov eax, ER_2 ; LOAD ERROR ADDRESS
2499 000049EF 50 <1> push eax ; PUSH NEC_OUT ERROR RETURN
2500 <1> ;mov ah, [ebp+1] ; OUTPUT TRACK #
2501 <1> ; 11/08/2022
2502 000049F0 8A642425 <1> mov ah, [esp+37] ; CH = OUTPUT TRACK #
2503 000049F4 E809030000 <1> call NEC_OUTPUT
2504 <1> ;mov ax, si ; OUTPUT HEAD #
2505 <1> ; 06/08/2022
2506 000049F9 89F0 <1> mov eax, esi
2507 000049FB E802030000 <1> call NEC_OUTPUT
2508 <1> ;mov ah, [ebp] ; OUTPUT SECTOR #
2509 <1> ; 11/08/2022
2510 00004A00 8A642424 <1> mov ah, [esp+36] ; CL = OUTPUT SECTOR #
2511 00004A04 E8F9020000 <1> call NEC_OUTPUT
2512 <1> ;mov dl, 3 ; BYTES/SECTOR PARAMETER FROM BLOCK
2513 <1> ; 06/08/2022
2514 00004A09 B003 <1> mov al, 3
2515 00004A0B E8FC010000 <1> call GET_PARM ; ... TO THE NEC
2516 00004A10 E8ED020000 <1> call NEC_OUTPUT ; OUTPUT TO CONTROLLER
2517 <1> ;mov dl, 4 ; EOT PARAMETER FROM BLOCK
2518 <1> ; 06/08/2022
2519 00004A15 B004 <1> mov al, 4
2520 00004A17 E8F0010000 <1> call GET_PARM ; ... TO THE NEC
2521 00004A1C E8E1020000 <1> call NEC_OUTPUT ; OUTPUT TO CONTROLLER
2522 00004A21 8A6305 <1> mov ah, [ebx+MD.GAP] ; GET GAP LENGTH
2523 <1> _R15:
2524 00004A24 E8D9020000 <1> call NEC_OUTPUT
2525 <1> ;mov dl, 6 ; DTL PARAMETER FROM BLOCK
2526 <1> ; 06/08/2022
2527 00004A29 B006 <1> mov al, 6
2528 00004A2B E8DC010000 <1> call GET_PARM ; ... TO THE NEC
2529 00004A30 E8CD020000 <1> call NEC_OUTPUT ; OUTPUT TO CONTROLLER
2530 00004A35 58 <1> pop eax ; THROW AWAY ERROR EXIT
2531 <1> ER_2:
2532 00004A36 C3 <1> ret
2533 <1>
2534 <1> ;-----
2535 <1> ; NEC_TERM
2536 <1> ; THIS ROUTINE WAITS FOR THE OPERATION THEN ACCEPTS THE STATUS
2537 <1> ; FROM THE NEC FOR THE READ/WRITE/VERIFY/FORWAT OPERATION.
2538 <1>
2539 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
2540 <1> ;-----
2541 <1> NEC_TERM:
2542 <1>
2543 <1> ;----- LET THE OPERATION HAPPEN
2544 <1>
2545 00004A37 56 <1> push esi ; SAVE HEAD #, # OF SECTORS
2546 00004A38 E8CD030000 <1> call WAIT_INT ; WAIT FOR THE INTERRUPT
2547 00004A3D 9C <1> pushf
2548 00004A3E E8F6030000 <1> call RESULTS ; GET THE NEC STATUS
2549 00004A43 724B <1> jc short SET_END_POP
2550 00004A45 9D <1> popf
2551 00004A46 723E <1> jc short SET_END ; LOOK FOR ERROR
2552 <1>
2553 <1> ;----- CHECK THE RESULTS RETURNED BY THE CONTROLLER
2554 <1>
2555 00004A48 FC <1> cld ; SET THE CORRECT DIRECTION
2556 00004A49 BE[29770100] <1> mov esi, NEC_STATUS ; POINT TO STATUS FIELD
2557 00004A4E AC <1> lodsb ; GET ST0
2558 00004A4F 24C0 <1> and AL, 11000000B ; TEST FOR NORMAL TERMINATION
2559 00004A51 7433 <1> jz short SET_END
2560 00004A53 3C40 <1> cmp AL, 01000000B ; TEST FOR ABNORMAL TERMINATION
2561 00004A55 7527 <1> jnz short J18 ; NOT ABNORMAL, BAD NEC
2562 <1>
2563 <1> ;----- ABNORMAL TERMINATION, FIND OUT WHY
2564 <1>
2565 00004A57 AC <1> lodsb ; GET ST1
2566 00004A58 D0E0 <1> sal al, 1 ; TEST FOR EDT FOUND
2567 00004A5A B404 <1> mov ah, RECORD_NOT_FND
2568 00004A5C 7222 <1> jc short J19
2569 00004A5E C0E002 <1> sal al, 2
2570 00004A61 B410 <1> mov ah, BAD_CRC
2571 00004A63 721B <1> jc short J19
2572 00004A65 D0E0 <1> sal al, 1 ; TEST FOR DMA OVERRUN
2573 00004A67 B408 <1> mov ah, BAD_DMA
2574 00004A69 7215 <1> jc short J19
2575 00004A6B C0E002 <1> sal al, 2 ; TEST FOR RECORD NOT FOUND
2576 00004A6E B404 <1> mov ah, RECORD_NOT_FND
2577 00004A70 720E <1> jc short J19
2578 00004A72 D0E0 <1> sal al, 1
2579 00004A74 B403 <1> mov ah, WRITE_PROTECT ; TEST FOR WRITE_PROTECT
2580 00004A76 7208 <1> jc short J19
2581 00004A78 D0E0 <1> sal al, 1 ; TEST MISSING ADDRESS MARK

```

```

2582 00004A7A B402      <1>      mov     ah, BAD_ADDR_MARK
2583 00004A7C 7202      <1>      jc      short J19
2584                      <1>
2585                      <1> ;----- NEC MUST HAVE FAILED
2586                      <1> J18:
2587 00004A7E B420      <1>      mov     ah, BAD_NEC
2588                      <1> J19:
2589 00004A80 0825[28770100] <1>      or      [DSKETTE_STATUS], ah
2590                      <1> SET_END:
2591 00004A86 803D[28770100]01 <1>      cmp     byte [DSKETTE_STATUS], 1 ; SET ERROR CONDITION
2592 00004A8D F5        <1>      cmc
2593 00004A8E 5E        <1>      pop     esi
2594 00004A8F C3        <1>      retn                ; RESTORE HEAD #, # OF SECTORS
2595                      <1>
2596                      <1> SET_END_POP:
2597 00004A90 9D        <1>      popf
2598 00004A91 EBF3      <1>      jmp     short SET_END
2599                      <1>
2600                      <1> ;-----
2601                      <1> ; DSTATE: ESTABLISH STATE UPON SUCCESSFUL OPERATION.
2602                      <1> ;-----
2603                      <1> DSTATE:
2604 00004A93 803D[28770100]00 <1>      cmp     byte [DSKETTE_STATUS], 0 ; CHECK FOR ERROR
2605 00004A9A 753E      <1>      jnz     short SETBAC ; IF ERROR JUMP
2606 00004A9C 808F[33770100]10 <1>      or      byte [DSK_STATE+edi], MED_DET
2607                      <1> ; NO ERROR, MARK MEDIA AS DETERMINED
2608 00004AA3 F687[33770100]04 <1>      test    byte [DSK_STATE+edi], DRV_DET ; DRIVE DETERMINED ?
2609 00004AAA 752E      <1>      jnz     short SETBAC ; IF DETERMINED NO TRY TO DETERMINE
2610 00004AAC 8A87[33770100] <1>      mov     al, [DSK_STATE+edi] ; LOAD STATE
2611 00004AB2 24C0      <1>      and     al, RATE_MSK ; KEEP ONLY RATE
2612 00004AB4 3C80      <1>      cmp     al, RATE_250 ; RATE 250 ?
2613 00004AB6 751B      <1>      jne     short M_12 ; NO, MUST BE 1.2M OR 1.44M DRIVE
2614                      <1>
2615                      <1> ;----- CHECK IF IT IS 1.44M
2616                      <1>
2617 00004AB8 E846010000 <1>      call    CMOS_TYPE ; RETURN DRIVE TYPE IN (AL)
2618                      <1> ;;20/02/2015
2619                      <1> ;;jc      short M_12 ; CMOS BAD
2620 00004ABD 7414      <1>      jz      short M_12 ;; 20/02/2015
2621 00004ABF 3C04      <1>      cmp     al, 4 ; 1.44MB DRIVE ?
2622 00004AC1 7410      <1>      je      short M_12 ; YES
2623                      <1> M_720:
2624 00004AC3 80A7[33770100]FD <1>      and     byte [DSK_STATE+edi], ~FMT_CAPA ; TURN OFF FORMAT CAPABILITY
2625 00004ACA 808F[33770100]04 <1>      or      byte [DSK_STATE+edi], DRV_DET ; MARK DRIVE DETERMINED
2626 00004AD1 EB07      <1>      jmp     short SETBAC ; BACK
2627                      <1> M_12:
2628 00004AD3 808F[33770100]06 <1>      or      byte [DSK_STATE+edi], DRV_DET+FMT_CAPA
2629                      <1> ; TURN ON DETERMINED & FMT CAPA
2630                      <1>
2631 00004ADA C3        <1>      SETBAC:
2632                      <1>      retn
2633                      <1>
2634                      <1> ;-----
2635                      <1> ; RETRY
2636                      <1> ; DETERMINES WHETHER A RETRY IS NECESSARY.
2637                      <1> ; IF RETRY IS REQUIRED THEN STATE INFORMATION IS UPDATED FOR RETRY.
2638                      <1> ;
2639                      <1> ; ON EXIT: CY = 1 FOR RETRY, CY = 0 FOR NO RETRY
2640                      <1> ;-----
2641                      <1> RETRY:
2642                      <1> ; 06/08/2022 - TRDOS 386 v2.0.5
2643 00004ADB 803D[28770100]00 <1>      cmp     byte [DSKETTE_STATUS], 0 ; GET STATUS OF OPERATION
2644 00004AE2 7445      <1>      jz      short NO_RETRY ; SUCCESSFUL OPERATION
2645 00004AE4 803D[28770100]80 <1>      cmp     byte [DSKETTE_STATUS], TIME_OUT ; IF TIME OUT NO RETRY
2646 00004AEB 743C      <1>      jz      short NO_RETRY
2647 00004AED 8AA7[33770100] <1>      mov     ah, [DSK_STATE+edi] ; GET MEDIA STATE OF DRIVE
2648 00004AF3 F6C410    <1>      test    ah, MED_DET ; ESTABLISHED/DETERMINED ?
2649 00004AF6 7531      <1>      jnz     short NO_RETRY ; IF ESTABLISHED STATE THEN TRUE ERROR
2650 00004AF8 80E4C0    <1>      and     ah, RATE_MSK ; ISOLATE RATE
2651 00004AFB 8A2D[30770100] <1>      mov     ch, [LAstrate] ; GET START OPERATION STATE
2652 00004B01 C0C504    <1>      rol     ch, 4 ; TO CORRESPONDING BITS
2653 00004B04 80E5C0    <1>      and     ch, RATE_MSK ; ISOLATE RATE BITS
2654 00004B07 38E5      <1>      cmp     ch, ah ; ALL RATES TRIED
2655 00004B09 741E      <1>      je      short NO_RETRY ; IF YES, THEN TRUE ERROR
2656                      <1>
2657                      <1> ; SETUP STATE INDICATOR FOR RETRY ATTEMPT TO NEXT RATE
2658                      <1> ; 00000000B (500) -> 10000000B (250)
2659                      <1> ; 10000000B (250) -> 01000000B (300)
2660                      <1> ; 01000000B (300) -> 00000000B (500)
2661 00004B0B 80FC01    <1>      cmp     ah, RATE_500+1 ; SET CY FOR RATE 500
2662 00004B0E D0DC      <1>      rcr     ah, 1 ; TO NEXT STATE
2663 00004B10 80E4C0    <1>      and     ah, RATE_MSK ; KEEP ONLY RATE BITS
2664 00004B13 80A7[33770100]1F <1>      and     byte [DSK_STATE+edi], ~(RATE_MSK+DBL_STEP)
2665                      <1> ; RATE, DBL STEP OFF
2666 00004B1A 08A7[33770100] <1>      or      [DSK_STATE+edi], ah ; TURN ON NEW RATE
2667 00004B20 C605[28770100]00 <1>      mov     byte [DSKETTE_STATUS], 0 ; RESET STATUS FOR RETRY
2668 00004B27 F9        <1>      stc ; SET CARRY FOR RETRY
2669 00004B28 C3        <1>      retn ; RETRY RETURN
2670                      <1>
2671                      <1> NO_RETRY:
2672                      <1> ; 06/08/2022
2673                      <1> ;
2674 00004B29 C3        <1>      ;clc ; CLEAR CARRY NO RETRY
2675                      <1>      retn ; NO RETRY RETURN
2676                      <1>
2677                      <1> ;-----
2678                      <1> ; NUM_TRANS
2679                      <1> ; THIS ROUTINE CALCULATES THE NUMBER OF SECTORS THAT WERE
2680                      <1> ; ACTUALLY TRANSFERRED TO/FROM THE DISKETTE.
2681                      <1> ;
2682                      <1> ; ON ENTRY: [BP+1] = TRACK
2683                      <1> ; SI-HI = HEAD
2684                      <1> ; [BP] = START SECTOR
2685                      <1> ;
2686                      <1> ; ON EXIT: AL = NUMBER ACTUALLY TRANSFERRED
2687                      <1> ;-----
2688                      <1> NUM_TRANS:
2689                      <1> ; 10/08/2022
2690                      <1> ; 06/08/2022 - TRDOS 386 v2.0.5
2691 00004B2A 30C0      <1>      xor     al, al ; CLEAR FOR ERROR
2692 00004B2C 3805[28770100] <1>      cmp     byte [DSKETTE_STATUS], 0
2693 00004B32 752D      <1>      jnz     short NT_OUT ; IF ERROR 0 TRANSFERRED
2694                      <1> ;mov     dl, 4 ; SECTORS/TRACK OFFSET TO DL
2695                      <1> ; 06/08/2022
2696 00004B34 B004      <1>      mov     al, 4
2697 00004B36 E8D1000000 <1>      call    GET_PARM ; AH = SECTORS/TRACK
2698 00004B3B 8A1D[2E770100] <1>      mov     bl, [NEC_STATUS+5] ; GET ENDING SECTOR
2699                      <1> ;mov     cx, si ; CH = HEAD # STARTED
2700                      <1> ; 06/08/2022
2701 00004B41 89F1      <1>      mov     ecx, esi
2702 00004B43 3A2D[2D770100] <1>      cmp     ch, [NEC_STATUS+4] ; GET HEAD ENDED UP ON
2703 00004B49 750E      <1>      jnz     short DIF_HD ; IF ON SAME HEAD, THEN NO ADJUST
2704 00004B4B 8A2D[2C770100] <1>      mov     ch, [NEC_STATUS+3] ; GET TRACK ENDED UP ON
2705                      <1> ;cmp     ch, [ebp+1] ; IS IT ASKED FOR TRACK

```



```

2706      ; 10/08/2022
2707 00004B51 3A6C241D      <1>      cmp     ch, [esp+29] ; CH = TRACK #
2708 00004B55 7404      <1>      jz      short SAME_TRK      ; IF SAME TRACK NO INCREASE
2709 00004B57 00E3      <1>      add     b1, ah      ; ADD SECTORS/TRACK
2710      <1> DIF_HD:
2711 00004B59 00E3      <1>      add     b1, ah      ; ADD SECTORS/TRACK
2712      <1> SAME_TRK:
2713      <1>      ;sub     b1, [ebp]      ; SUBTRACT START FROM END
2714      <1>      ; 10/08/2022
2715 00004B5B 2A5C241C      <1>      sub     b1, [esp+28] ; CL = SECTOR #
2716 00004B5F 88D8      <1>      mov     al, b1      ; TO AL
2717      <1> NT_OUT:
2718 00004B61 C3      <1>      retn
2719      <1>
2720      <1> ;-----
2721      <1> ; SETUP_DBL
2722      <1> ; CHECK DOUBLE STEP.
2723      <1> ;
2724      <1> ; ON ENTRY :      DI = DRIVE
2725      <1> ;
2726      <1> ; ON EXIT : CY = 1 MEANS ERROR
2727      <1> ;-----
2728      <1> SETUP_DBL:
2729      <1> ; 06/08/2022 - TRDOS 386 v2.0.5
2730 00004B62 8AA7[33770100]      <1>      mov     ah, [DSK_STATE+edi] ; ACCESS STATE
2731 00004B68 F6C410      <1>      test    ah, MED_DET      ; ESTABLISHED STATE ?
2732 00004B6B 7578      <1>      jnz     short NO_DBL      ; IF ESTABLISHED THEN DOUBLE DONE
2733      <1>
2734      <1> ;----- CHECK FOR TRACK 0 TO SPEED UP ACKNOWLEDGE OF UNFORMATTED DISKETTE
2735      <1>
2736 00004B6D C605[25770100]00      <1>      mov     byte [SEEK_STATUS], 0 ; SET RECALIBRATE REQUIRED ON ALL DRIVES
2737 00004B74 E8CE000000      <1>      call    MOTOR_ON      ; ENSURE MOTOR STAY ON
2738 00004B79 B500      <1>      mov     ch, 0      ; LOAD TRACK 0
2739 00004B7B E8B9010000      <1>      call    SEEK      ; SEEK TO TRACK 0
2740 00004B80 E861000000      <1>      call    READ_ID      ; READ ID FUNCTION
2741 00004B85 7243      <1>      jc      short SD_ERR      ; IF ERROR NO TRACK 0
2742      <1>
2743      <1> ;----- INITIALIZE START AND MAX TRACKS (TIMES 2 FOR BOTH HEADS)
2744      <1>
2745 00004B87 66B95004      <1>      mov     cx, 0450h      ; START, MAX TRACKS
2746 00004B8B F687[33770100]01      <1>      test    byte [DSK_STATE+edi], TRK_CAPA ; TEST FOR 80 TRACK CAPABILITY
2747 00004B92 7402      <1>      jz      short CNT_OK      ; IF NOT COUNT IS SETUP
2748 00004B94 B1A0      <1>      mov     c1, 0A0h      ; MAXIMUM TRACK 1.2 MB
2749      <1>
2750      <1> ; ATTEMPT READ ID OF ALL TRACKS, ALL HEADS UNTIL SUCCESS; UPON SUCCESS,
2751      <1> ; MUST SEE IF ASKED FOR TRACK IN SINGLE STEP MODE = TRACK ID READ; IF NOT
2752      <1> ; THEN SET DOUBLE STEP ON.
2753      <1>
2754      <1> CNT_OK:
2755      <1> ; 11/04/2021 (32 bit push/pop)
2756 00004B96 C605[27770100]FF      <1>      mov     byte [MOTOR_COUNT], 0FFh ; ENSURE MOTOR STAYS ON FOR OPERATION
2757 00004B9D 51      <1>      push    ecx      ; SAVE TRACK, COUNT
2758 00004B9E C605[28770100]00      <1>      mov     byte [DSKETTE_STATUS], 0 ; CLEAR STATUS, EXPECT ERRORS
2759      <1>      ;xor     ax, ax      ; CLEAR AX
2760      <1> ; 06/08/2022
2761 00004BA5 31C0      <1>      xor     eax, eax
2762 00004BA7 D0ED      <1>      shr     ch, 1      ; HALVE TRACK, CY = HEAD
2763 00004BA9 C0D003      <1>      rcl     al, 3      ; AX = HEAD IN CORRECT BIT
2764 00004BAC 50      <1>      push    eax      ; SAVE HEAD
2765 00004BAD E887010000      <1>      call    SEEK      ; SEEK TO TRACK
2766 00004BB2 58      <1>      pop     eax      ; RESTORE HEAD
2767      <1>      ;or     di, ax      ; DI = HEAD OR'ED DRIVE
2768      <1> ; 06/08/2022
2769 00004BB3 09C7      <1>      or      edi, eax
2770 00004BB5 E82C000000      <1>      call    READ_ID      ; READ ID HEAD 0
2771 00004BBA 9C      <1>      pushf    ; SAVE RETURN FROM READ_ID
2772 00004BBB 6681E7FB00      <1>      and     di, 11111011b      ; TURN OFF HEAD 1 BIT
2773 00004BC0 9D      <1>      popf     ; RESTORE ERROR RETURN
2774 00004BC1 59      <1>      pop     ecx      ; RESTORE COUNT
2775 00004BC2 7308      <1>      jnc     short DO_CHK      ; IF OK, ASKED = RETURNED TRACK ?
2776 00004BC4 FEC5      <1>      inc     ch      ; INC FOR NEXT TRACK
2777 00004BC6 38CD      <1>      cmp     ch, c1      ; REACHED MAXIMUM YET
2778 00004BC8 75CC      <1>      jnz     short CNT_OK      ; CONTINUE TILL ALL TRIED
2779      <1>
2780      <1> ;----- FALL THRU, READ ID FAILED FOR ALL TRACKS
2781      <1>
2782      <1> SD_ERR:
2783 00004BCA F9      <1>      stc      ; SET CARRY FOR ERROR
2784 00004BCB C3      <1>      retn      ; SETUP_DBL ERROR EXIT
2785      <1>
2786      <1> DO_CHK:
2787 00004BCC 8A0D[2C770100]      <1>      mov     c1, [NEC_STATUS+3] ; LOAD RETURNED TRACK
2788 00004BD2 888F[35770100]      <1>      mov     [DSK_TRK+edi], c1 ; STORE TRACK NUMBER
2789 00004BD8 D0ED      <1>      shr     ch, 1      ; HALVE TRACK
2790 00004BDA 38CD      <1>      cmp     ch, c1      ; IS IT THE SAME AS ASKED FOR TRACK
2791 00004BDC 7407      <1>      jz      short NO_DBL      ; IF SAME THEN NO DOUBLE STEP
2792 00004BDE 808F[33770100]20      <1>      or      byte [DSK_STATE+edi], DBL_STEP ; TURN ON DOUBLE STEP REQUIRED
2793      <1> NO_DBL:
2794      <1> ; 06/08/2022
2795      <1> ; c1c      ; CLEAR ERROR FLAG
2796 00004BE5 C3      <1>      retn
2797      <1>
2798      <1> ;-----
2799      <1> ; READ_ID
2800      <1> ; READ ID FUNCTION.
2801      <1> ;
2802      <1> ; ON ENTRY: DI : BIT 2 = HEAD; BITS 1,0 = DRIVE
2803      <1> ;
2804      <1> ; ON EXIT: DI : BIT 2 IS RESET, BITS 1,0 = DRIVE
2805      <1> ; @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION
2806      <1> ;-----
2807      <1> READ_ID:
2808      <1> ; 06/08/2022 - TRDOS 386 v2.0.5
2809 00004BE6 B8[024C0000]      <1>      mov     eax, ER_3      ; MOVE NEC OUTPUT ERROR ADDRESS
2810 00004BEB 50      <1>      push    eax
2811 00004BEC B44A      <1>      mov     ah, 4Ah      ; READ ID COMMAND
2812 00004BEE E80F010000      <1>      call    NEC_OUTPUT      ; TO CONTROLLER
2813      <1>      ;mov     ax, di      ; DRIVE # TO AH, HEAD 0
2814      <1> ; 06/08/2022
2815 00004BF3 89F8      <1>      mov     eax, edi
2816 00004BF5 88C4      <1>      mov     ah, al
2817 00004BF7 E806010000      <1>      call    NEC_OUTPUT      ; TO CONTROLLER
2818 00004BFC E836FEFFFF      <1>      call    NEC_TERM      ; WAIT FOR OPERATION, GET STATUS
2819 00004C01 58      <1>      pop     eax      ; THROW AWAY ERROR ADDRESS
2820      <1> ER_3:
2821 00004C02 C3      <1>      retn
2822      <1>
2823      <1> ;-----
2824      <1> ; CMOS_TYPE
2825      <1> ; RETURNS DISKETTE TYPE FROM CMOS
2826      <1> ;
2827      <1> ; ON ENTRY: DI = DRIVE #
2828      <1> ;
2829      <1> ; ON EXIT: AL = TYPE; CY REFLECTS STATUS

```

```

2830 <1> ;-----
2831 <1>
2832 <1> CMOS_TYPE: ; 11/12/2014
2833 00004C03 8A87[7C660000] <1> mov al, [edi+fd0_type]
2834 00004C09 20C0 <1> and al, al ; 18/12/2014
2835 00004C0B C3 <1> retn
2836 <1>
2837 <1> ;CMOS_TYPE:
2838 <1> ; mov al, CMOS_DIAG ; CMOS DIAGNOSTIC STATUS BYTE ADDRESS
2839 <1> ; call CMOS_READ ; GET CMOS STATUS
2840 <1> ; test al, BAD_BAT+BAD_CKSUM ; BATTERY GOOD AND CHECKSUM VALID
2841 <1> ; stc ; SET CY = 1 INDICATING ERROR FOR RETURN
2842 <1> ; jnz short BAD_CM ; ERROR IF EITHER BIT ON
2843 <1> ; mov al, CMOS_DISKETTE ; ADDRESS OF DISKETTE BYTE IN CMOS
2844 <1> ; call CMOS_READ ; GET DISKETTE BYTE
2845 <1> ; or di, di ; SEE WHICH DRIVE IN QUESTION
2846 <1> ; jnz short TB ; IF DRIVE 1, DATA IN LOW NIBBLE
2847 <1> ; ror al, 4 ; EXCHANGE NIBBLES IF SECOND DRIVE
2848 <1> ;TB:
2849 <1> ; and al, 0Fh ; KEEP ONLY DRIVE DATA, RESET CY, 0
2850 <1> ;BAD_CM:
2851 <1> ; retn ; CY, STATUS OF READ
2852 <1>
2853 <1> ;-----
2854 <1> ; GET_PARM
2855 <1> ; THIS ROUTINE FETCHES THE INDEXED POINTER FROM THE DISK_BASE
2856 <1> ; BLOCK POINTED TO BY THE DATA VARIABLE @DISK_POINTER. A BYTE FROM
2857 <1> ; THAT TABLE IS THEN MOVED INTO AH, THE INDEX OF THAT BYTE BEING
2858 <1> ; THE PARAMETER IN DL.
2859 <1> ;
2860 <1> ; ON ENTRY: DL = INDEX OF BYTE TO BE FETCHED
2861 <1> ;
2862 <1> ; ON EXIT: AH = THAT BYTE FROM BLOCK
2863 <1> ; AL, DH DESTROYED
2864 <1> ;-----
2865 <1> GET_PARM:
2866 <1> ; 09/08/2022
2867 <1> ; 06/08/2022 - TRDOS 386 v2.0.5
2868 <1> ; AL = INDEX
2869 <1> ; EDI = (current) DRIVE #
2870 <1> ; push ds
2871 <1> ; push esi
2872 <1> ; sub ax, ax ; DS = 0, BIOS DATA AREA
2873 <1> ; mov ds, ax
2874 <1> ; mov ax, cs
2875 <1> ; mov ds, ax
2876 <1> ; 08/02/2015 (protected mode modifications, bx -> ebx)
2877 <1> ; xchg edx, ebx ; BL = INDEX
2878 <1> ; 06/08/2022
2879 00004C0C 53 <1> push ebx ; SAVE EBX
2880 <1> ; movzx ebx, dl ; EBX = INDEX
2881 00004C0D 0FB6D8 <1> movzx ebx, al ; 06/08/2022
2882 <1> ; sub bh, bh ; BX = INDEX
2883 <1> ; and ebx, 0FFh
2884 <1> ; lds si, [DISK_POINTER] ; POINT TO BLOCK
2885 <1> ;
2886 <1> ; 17/12/2014
2887 <1> ; mov ax, [cfd] ; current (AL) and previous fd (AH)
2888 <1> ; 06/08/2022
2889 00004C10 89F8 <1> mov eax, edi ; EDI = DRIVE #
2890 <1> ; cmp al, ah
2891 00004C12 3A05[6D660000] <1> cmp al, [pfd]
2892 00004C18 7423 <1> je short gpndc
2893 00004C1A A2[6D660000] <1> mov [pfd], al ; current drive -> previous drive
2894 00004C1F 53 <1> push ebx ; 08/02/2015
2895 <1> ; mov bl, al
2896 <1> ; 11/12/2014
2897 <1> ; mov al, [ebx+fd0_type] ; Drive type (0,1,2,3,4)
2898 <1> ; 09/08/2022
2899 00004C20 8A87[7C660000] <1> mov al, [edi+fd0_type] ; Drive type (0,1,2,3,4)
2900 <1> ; 18/12/2014
2901 00004C26 20C0 <1> and al, al
2902 00004C28 7507 <1> jnz short gpdtc
2903 00004C2A 8B[57660000] <1> mov ebx, MD_TBL6 ; 1.44 MB param. tbl. (default)
2904 00004C2F EB05 <1> jmp short gpdpu
2905 <1> gpdtc:
2906 00004C31 E809FBFFFF <1> call DR_TYPE_CHECK
2907 <1> ; cf = 1 -> ebx points to 1.44MB fd parameter table (default)
2908 <1> gpdpu:
2909 00004C36 891D[F4650000] <1> mov [DISK_POINTER], ebx
2910 00004C3C 5B <1> pop ebx
2911 <1> gpndc:
2912 <1> ; mov esi, [DISK_POINTER] ; 08/02/2015, si -> esi
2913 <1> ; 06/08/2022
2914 00004C3D 031D[F4650000] <1> add ebx, [DISK_POINTER]
2915 <1> ; mov ah, [esi+ebx] ; GET THE WORD
2916 00004C43 8A23 <1> mov ah, [ebx]
2917 <1> ; xchg edx, ebx ; RESTORE BX
2918 <1> ; 06/08/2022
2919 00004C45 5B <1> pop ebx ; RESTORE EBX
2920 <1> ; pop esi
2921 <1> ; pop ds
2922 00004C46 C3 <1> retn
2923 <1>
2924 <1> ;-----
2925 <1> ; MOTOR_ON
2926 <1> ; TURN MOTOR ON AND WAIT FOR MOTOR START UP TIME. THE @MOTOR_COUNT
2927 <1> ; IS REPLACED WITH A SUFFICIENTLY HIGH NUMBER (0FFH) TO ENSURE
2928 <1> ; THAT THE MOTOR DOES NOT GO OFF DURING THE OPERATION. IF THE
2929 <1> ; MOTOR NEEDED TO BE TURNED ON, THE MULTI-TASKING HOOK FUNCTION
2930 <1> ; (AX=90FDH, INT 15) IS CALLED TELLING THE OPERATING SYSTEM
2931 <1> ; THAT THE BIOS IS ABOUT TO WAIT FOR MOTOR START UP. IF THIS
2932 <1> ; FUNCTION RETURNS WITH CY = 1, IT MEANS THAT THE MINIMUM WAIT
2933 <1> ; HAS BEEN COMPLETED. AT THIS POINT A CHECK IS MADE TO ENSURE
2934 <1> ; THAT THE MOTOR WASN'T TURNED OFF BY THE TIMER. IF THE HOOK DID
2935 <1> ; NOT WAIT, THE WAIT FUNCTION (AH=086H) IS CALLED TO WAIT THE
2936 <1> ; PRESCRIBED AMOUNT OF TIME. IF THE CARRY FLAG IS SET ON RETURN,
2937 <1> ; IT MEANS THAT THE FUNCTION IS IN USE AND DID NOT PERFORM THE
2938 <1> ; WAIT. A TIMER 1 WAIT LOOP WILL THEN DO THE WAIT.
2939 <1> ;
2940 <1> ; ON ENTRY: DI = DRIVE #
2941 <1> ; ON EXIT: AX,CX,DX DESTROYED
2942 <1> ;-----
2943 <1> MOTOR_ON:
2944 <1> ; 06/08/2022 - TRDOS 386 kernel v2.0.5
2945 00004C47 53 <1> push ebx ; SAVE REG.
2946 00004C48 E825000000 <1> call TURN_ON ; TURN ON MOTOR
2947 00004C4D 7221 <1> jc short MOT_IS_ON ; IF CY=1 NO WAIT
2948 <1> ; 06/08/2022
2949 <1> ; call XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
2950 00004C4F E851FBFFFF <1> call XLAT_NEW ; TRANSLATE STATE TO PRESENT ARCH,
2951 <1> ; call TURN_ON ; CHECK AGAIN IF MOTOR ON
2952 <1> ; jc short MOT_IS_ON ; IF NO WAIT MEANS IT IS ON
2953 <1> M_WAIT:

```

```

2954      <1>      ;mov     dl, 10                ; GET THE MOTOR WAIT PARAMETER
2955      <1>      ; 06/08/2022
2956      00004C54 B00A      <1>      mov     al, 10
2957      00004C56 E8B1FFFFFF <1>      call    GET_PARM
2958      <1>      ;mov     al, ah                ; AL = MOTOR WAIT PARAMETER
2959      <1>      ;xor     ah, ah                ; AX = MOTOR WAIT PARAMETER
2960      <1>      ;cmp     al, 8                ; SEE IF AT LEAST A SECOND IS SPECIFIED
2961      00004C5B 80FC08     <1>      cmp     ah, 8
2962      <1>      ;jae     short GP2            ; IF YES, CONTINUE
2963      00004C5E 7702      <1>      ja      short J13
2964      <1>      ;mov     al, 8                ; ONE SECOND WAIT FOR MOTOR START UP
2965      00004C60 B408      <1>      mov     ah, 8
2966      <1>
2967      <1>      ;----- AS CONTAINS NUMBER OF 1/8 SECONDS (125000 MICROSECONDS) TO WAIT
2968      GP2:      <1>      ;----- FOLLOWING LOOPS REQUIRED WHEN RTC WAIT FUNCTION IS ALREADY IN USE
2969      <1>      J13:      <1>      ; WAIT FOR 1/8 SECOND PER (AL)
2970      <1>      <1>      ; COUNT FOR 1/8 SECOND AT 15.085737 US
2971      <1>      ;mov     ecx, 8286
2972      <1>      ; 11/04/2021
2973      00004C62 B947100000 <1>      mov     ecx, 4167 ; count of 30 micro seconds * (1/8)
2974      00004C67 E8B5D7FFFF <1>      call    WAITF      ; GO TO FIXED WAIT ROUTINE
2975      <1>      ;dec     al                    ; DECREMENT TIME VALUE
2976      00004C6C FECC      <1>      dec     ah
2977      00004C6E 75F2      <1>      jnz     short J13      ; ARE WE DONE YET
2978      <1>      MOT_IS_ON:
2979      00004C70 5B      <1>      pop     ebx      ; RESTORE REG.
2980      00004C71 C3      <1>      retn
2981      <1>
2982      <1>      ;-----
2983      <1>      ; TURN_ON
2984      <1>      ; TURN MOTOR ON AND RETURN WAIT STATE.
2985      <1>
2986      <1>      ; ON ENTRY: DI = DRIVE #
2987      <1>
2988      <1>      ; ON EXIT: CY = 0 MEANS WAIT REQUIRED
2989      <1>      ; CY = 1 MEANS NO WAIT REQUIRED
2990      <1>      ; AX,BX,CX,DX DESTROYED
2991      <1>      ;-----
2992      <1>      TURN_ON:
2993      00004C72 89FB      <1>      mov     ebx, edi      ; BX = DRIVE #
2994      00004C74 88D9      <1>      mov     cl, bl      ; CL = DRIVE #
2995      00004C76 C0C304    <1>      rol     bl, 4      ; BL = DRIVE SELECT
2996      00004C79 FA      <1>      cli      ; NO INTERRUPTS WHILE DETERMINING STATUS
2997      00004C7A C605[26770100]FF <1>      mov     byte [MOTOR_COUNT], 0FFh ; ENSURE MOTOR STAYS ON FOR OPERATION
2998      00004C81 A0[26770100] <1>      mov     al, [MOTOR_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION
2999      00004C86 2430      <1>      and     al, 00110000b ; KEEP ONLY DRIVE SELECT BITS
3000      00004C88 B401      <1>      mov     ah, 1      ; MASK FOR DETERMINING MOTOR BIT
3001      00004C8A D2E4      <1>      shl     ah, cl      ; AH = MOTOR ON, A=00000001, B=00000010
3002      <1>
3003      <1>      ; AL = DRIVE SELECT FROM @MOTOR_STATUS
3004      <1>      ; BL = DRIVE SELECT DESIRED
3005      <1>      ; AH = MOTOR ON MASK DESIRED
3006      <1>
3007      00004C8C 38D8      <1>      cmp     al, bl      ; REQUESTED DRIVE ALREADY SELECTED ?
3008      00004C8E 7508      <1>      jnz     short TURN_IT_ON ; IF NOT SELECTED JUMP
3009      00004C90 8425[26770100] <1>      test    ah, [MOTOR_STATUS] ; TEST MOTOR ON BIT
3010      00004C96 7535      <1>      jnz     short NO_MOT_WAIT ; JUMP IF MOTOR ON AND SELECTED
3011      <1>
3012      <1>      TURN_IT_ON:
3013      00004C98 08DC      <1>      or      ah, bl      ; AH = DRIVE SELECT AND MOTOR ON
3014      00004C9A 8A3D[26770100] <1>      mov     bh, [MOTOR_STATUS] ; SAVE COPY OF @MOTOR_STATUS BEFORE
3015      00004CA0 80E70F    <1>      and     bh, 00001111b ; KEEP ONLY MOTOR BITS
3016      00004CA3 8025[26770100]CF <1>      and     byte [MOTOR_STATUS], 11001111b ; CLEAR OUT DRIVE SELECT
3017      00004CAA 0825[26770100] <1>      or      [MOTOR_STATUS], ah ; OR IN DRIVE SELECTED AND MOTOR ON
3018      00004CB0 A0[26770100] <1>      mov     al, [MOTOR_STATUS] ; GET DIGITAL OUTPUT REGISTER REFLECTION
3019      00004CB5 88C3      <1>      mov     bl, al      ; BL=@MOTOR_STATUS AFTER, BH=BEFORE
3020      00004CB7 80E30F    <1>      and     bl, 00001111b ; KEEP ONLY MOTOR BITS
3021      00004CBA FB      <1>      sti      ; ENABLE INTERRUPTS AGAIN
3022      00004CBB 243F      <1>      and     al, 00111111b ; STRIP AWAY UNWANTED BITS
3023      00004CBD C0C004    <1>      rol     al, 4      ; PUT BITS IN DESIRED POSITIONS
3024      00004CC0 0C0C      <1>      or      al, 00001100b ; NO RESET, ENABLE DMA/INTERRUPT
3025      00004CC2 66BAF203 <1>      mov     dx, 03F2h ; SELECT DRIVE AND TURN ON MOTOR
3026      00004CC6 EE      <1>      out     dx, al
3027      00004CC7 38FB      <1>      cmp     bl, bh      ; NEW MOTOR TURNED ON ?
3028      <1>      ;jz      short NO_MOT_WAIT ; NO WAIT REQUIRED IF JUST SELECT
3029      00004CC9 7403      <1>      je      short no_mot_w1 ; 27/02/2015
3030      00004CCB F8      <1>      clc      ; (re)SET CARRY MEANING WAIT
3031      00004CCC C3      <1>      retn
3032      <1>
3033      <1>      NO_MOT_WAIT:
3034      00004CCD FB      <1>      sti      ; INTERRUPTS BACK ON
3035      <1>      no_mot_w1: ; 27/02/2015
3036      00004CCE F9      <1>      stc      ; SET NO WAIT REQUIRED
3037      <1>      ;sti
3038      00004CCF C3      <1>      retn
3039      <1>
3040      <1>      ;-----
3041      <1>      ; HD_WAIT
3042      <1>      ; WAIT FOR HEAD SETTLE TIME.
3043      <1>
3044      <1>      ; ON ENTRY: DI = DRIVE #
3045      <1>
3046      <1>      ; ON EXIT: AX,BX,CX,DX DESTROYED
3047      <1>      ;-----
3048      <1>      HD_WAIT:
3049      <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
3050      <1>      ;mov     dl, 9                ; GET HEAD SETTLE PARAMETER
3051      <1>      ; 06/08/2022
3052      00004CD0 B009      <1>      mov     al, 9
3053      00004CD2 E835FFFFFF <1>      call    GET_PARM
3054      00004CD7 08E4      <1>      or      ah, ah ; 17/12/2014
3055      00004CD9 7519      <1>      jnz     short DO_WAT
3056      00004CDB F605[26770100]80 <1>      test    byte [MOTOR_STATUS], 10000000b ; SEE IF A WRITE OPERATION
3057      <1>      ;jz      short ISNT_WRITE ; IF NOT, DO NOT ENFORCE ANY VALUES
3058      <1>      ;or      ah, ah                ; CHECK FOR ANY WAIT?
3059      <1>      ;jnz     short DO_WAT            ; IF THERE DO NOT ENFORCE
3060      00004CE2 741D      <1>      jz      short HW_DONE
3061      00004CE4 B40F      <1>      mov     ah, HD12_SETTLE ; LOAD 1.2M HEAD SETTLE MINIMUM
3062      00004CE6 8A87[33770100] <1>      mov     al, [DSK_STATE+edi] ; LOAD STATE
3063      00004CEC 24C0      <1>      and     al, RATE_MSK ; KEEP ONLY RATE
3064      00004CEE 3C80      <1>      cmp     al, RATE_250 ; 1.2 M DRIVE ?
3065      00004CF0 7502      <1>      jnz     short DO_WAT ; DEFAULT HEAD SETTLE LOADED
3066      <1>      ;GP3:
3067      00004CF2 B414      <1>      mov     ah, HD320_SETTLE ; USE 320/360 HEAD SETTLE
3068      <1>      ; jmp     short DO_WAT
3069      <1>
3070      <1>      ;ISNT_WRITE:
3071      <1>      ; or      ah, ah                ; CHECK FOR NO WAIT
3072      <1>      ; jz      short HW_DONE            ; IF NOT WRITE AND 0 ITS OK
3073      <1>
3074      <1>      ;----- AH CONTAINS NUMBER OF MILLISECONDS TO WAIT
3075      <1>      DO_WAT:
3076      <1>      ; mov     al, ah                ; AL = # MILLISECONDS
3077      <1>      ; ;xor     ah, ah                ; AX = # MILLISECONDS

```

```

3078 <1> J29: ; 1 MILLISECOND LOOP
3079 <1> ;mov cx, WAIT_FDU_HEAD_SETTLE ; 33 ; 1 ms in 30 micro units.
3080 <1> ;mov ecx, 66 ; COUNT AT 15.085737 US PER COUNT
3081 <1> ; 11/04/2021
3082 <1> ;mov ecx, WAIT_FDU_HEAD_SETTLE ; 33
3083 <1> ; 06/08/2022
3084 00004CF4 29C9 <1> sub ecx, ecx
3085 00004CF6 B121 <1> mov cl, WAIT_FDU_HEAD_SETTLE ; 33
3086 00004CF8 E824D7FFFF <1> call WAITF ; DELAY FOR 1 MILLISECOND
3087 <1> ;dec al ; DECREMENT THE COUNT
3088 00004CFD FECC <1> dec ah
3089 00004CFF 75F3 <1> jnz short J29 ; DO AL MILLISECOND # OF TIMES
3090 <1> HW_DONE:
3091 00004D01 C3 <1> retn
3092 <1>
3093 <1> -----
3094 <1> ; NEC_OUTPUT
3095 <1> ; THIS ROUTINE SENDS A BYTE TO THE NEC CONTROLLER AFTER TESTING
3096 <1> ; FOR CORRECT DIRECTION AND CONTROLLER READY THIS ROUTINE WILL
3097 <1> ; TIME OUT IF THE BYTE IS NOT ACCEPTED WITHIN A REASONABLE AMOUNT
3098 <1> ; OF TIME, SETTING THE DISKETTE STATUS ON COMPLETION.
3099 <1> ;
3100 <1> ; ON ENTRY: AH = BYTE TO BE OUTPUT
3101 <1> ;
3102 <1> ; ON EXIT: CY = 0 SUCCESS
3103 <1> ; CY = 1 FAILURE -- DISKETTE STATUS UPDATED
3104 <1> ; IF A FAILURE HAS OCCURRED, THE RETURN IS MADE ONE LEVEL
3105 <1> ; HIGHER THAN THE CALLER OF NEC_OUTPUT. THIS REMOVES THE
3106 <1> ; REQUIREMENT OF TESTING AFTER EVERY CALL OF NEC_OUTPUT.
3107 <1> ; AX,CX,DX DESTROYED
3108 <1> -----
3109 <1>
3110 <1> ; 09/12/2014 [Erdogan Tan]
3111 <1> ; (from 'PS2 Hardware Interface Tech. Ref. May 88', Page 09-05.)
3112 <1> ; Diskette Drive Controller Status Register (3F4h)
3113 <1> ; This read only register facilitates the transfer of data between
3114 <1> ; the system microprocessor and the controller.
3115 <1> ; Bit 7 - When set to 1, the Data register is ready to transfer data
3116 <1> ; with the system microprocessor.
3117 <1> ; Bit 6 - The direction of data transfer. If this bit is set to 0,
3118 <1> ; the transfer is to the controller.
3119 <1> ; Bit 5 - When this bit is set to 1, the controller is in the non-DMA mode.
3120 <1> ; Bit 4 - When this bit is set to 1, a Read or Write command is being executed.
3121 <1> ; Bit 3 - Reserved.
3122 <1> ; Bit 2 - Reserved.
3123 <1> ; Bit 1 - When this bit is set to 1, diskette drive 1 is in the seek mode.
3124 <1> ; Bit 0 - When this bit is set to 1, diskette drive 1 is in the seek mode.
3125 <1>
3126 <1> ; Data Register (3F5h)
3127 <1> ; This read/write register passes data, commands and parameters, and provides
3128 <1> ; diskette status information.
3129 <1>
3130 <1> NEC_OUTPUT:
3131 <1> ; 09/08/2022
3132 <1> ; 06/08/2022 - TRDOS 386 Kernel v2.0.5
3133 <1> ;
3134 <1> ;push bx ; SAVE REG.
3135 00004D02 66BAF403 <1> mov dx, 03F4h ; STATUS PORT
3136 <1> ;mov bl, 2 ; HIGH ORDER COUNTER
3137 <1> ;xor cx, cx ; COUNT FOR TIME OUT
3138 <1> ; 16/12/2014
3139 <1> ; waiting for (max.) 0.5 seconds
3140 <1> ;mov byte [wait_count], 0 ; 27/02/2015
3141 <1> ;
3142 <1> ; 17/12/2014
3143 <1> ; Modified from AWARD BIOS 1999 - ADISK.ASM - SEND_COMMAND
3144 <1> ;
3145 <1> ;WAIT_FOR_PORT: Waits for a bit at a port pointed to by DX to
3146 <1> ; go on.
3147 <1> ;INPUT:
3148 <1> ; AH=Mask for isolation bits.
3149 <1> ; AL=pattern to look for.
3150 <1> ; DX=Port to test for
3151 <1> ; BH:CX=Number of memory refresh periods to delay.
3152 <1> ; (normally 30 microseconds per period.)
3153 <1> ;
3154 <1> ;WFP_SHORT:
3155 <1> ; Wait for port if refresh cycle is short (15-80 us range).
3156 <1> ;
3157 <1> ;
3158 <1> ; mov bl, WAIT_FDU_SEND_HI+1; 0+1
3159 <1> ; mov cx, WAIT_FDU_SEND_LO ; 16667
3160 00004D06 B91B410000 <1> mov ecx, WAIT_FDU_SEND_LH ; 16667 (27/02/2015)
3161 <1> ;
3162 <1> ;WFPS_OUTER_LP:
3163 <1> ;
3164 <1> ;WFPS_CHECK_PORT:
3165 <1> J23:
3166 <1> in al, dx ; GET STATUS
3167 00004D0C 24C0 <1> and al, 11000000b ; KEEP STATUS AND DIRECTION
3168 <1> cmp al, 10000000b ; STATUS 1 AND DIRECTION 0 ?
3169 00004D10 7418 <1> jz short J27 ; STATUS AND DIRECTION OK
3170 <1> WFPS_HI:
3171 <1> in al, PORT_B ; 061h ; SYS1 ; wait for hi to lo
3172 00004D14 A810 <1> test al, 010h ; transition on memory
3173 00004D16 75FA <1> jnz short WFPS_HI ; refresh.
3174 <1> WFPS_LO:
3175 00004D18 E461 <1> in al, PORT_B ; SYS1
3176 00004D1A A810 <1> test al, 010h
3177 00004D1C 74FA <1> jz short WFPS_LO
3178 <1> ;loop short WFPS_CHECK_PORT
3179 00004D1E E2EB <1> loop J23 ; 27/02/2015
3180 <1> ;
3181 <1> ; dec bl
3182 <1> ; jnz short WFPS_OUTER_LP
3183 <1> ; jmp short WFPS_TIMEOUT ; fail
3184 <1> ;J23:
3185 <1> in al, dx ; GET STATUS
3186 <1> and al, 11000000b ; KEEP STATUS AND DIRECTION
3187 <1> cmp al, 10000000b ; STATUS 1 AND DIRECTION 0 ?
3188 <1> ; jz short J27 ; STATUS AND DIRECTION OK
3189 <1> ;loop J23 ; CONTINUE TILL CX EXHAUSTED
3190 <1> ;dec bl ; DECREMENT COUNTER
3191 <1> ;jnz short J23 ; REPEAT TILL DELAY FINISHED, CX = 0
3192 <1> ;
3193 <1> ; 27/02/2015
3194 <1> ; 16/12/2014
3195 <1> ;cmp byte [wait_count], 10 ; (10/18.2 seconds)
3196 <1> ;jnb short J23
3197 <1> ;
3198 <1> ;WFPS_TIMEOUT:
3199 <1> ;
3200 <1> ;----- FALL THRU TO ERROR RETURN
3201 <1>

```

```

3202 00004D20 800D[28770100]80 <1> or byte [DSKETTE_STATUS], TIME_OUT
3203 <1> ;pop bx ; RESTORE REG.
3204 00004D27 58 <1> pop eax ; 08/02/2015 ; DISCARD THE RETURN ADDRESS
3205 00004D28 F9 <1> stc ; INDICATE ERROR TO CALLER
3206 00004D29 C3 <1> retn
3207 <1>
3208 <1> ;----- DIRECTION AND STATUS OK; OUTPUT BYTE
3209 <1>
3210 <1> J27:
3211 00004D2A 88E0 <1> mov al, ah ; GET BYTE TO OUTPUT
3212 <1> ;inc dx ; DATA PORT = STATUS PORT + 1
3213 <1> ; 06/08/2022
3214 00004D2C FEC2 <1> inc dl
3215 00004D2E EE <1> out dx, al ; OUTPUT THE BYTE
3216 <1> ;;NEWIODELAY ;; 27/02/2015
3217 <1> ; 27/02/2015
3218 <1> ;pushf ; SAVE FLAGS
3219 <1> ; 09/08/2022
3220 <1> ; cf = 0, zf = 1
3221 <1> ;mov ecx, 3 ; 30 TO 45 MICROSECONDS WAIT FOR
3222 <1> ; 11/04/2021
3223 <1> ;mov ecx, 2
3224 <1> ; 06/08/2022
3225 00004D2F 29C9 <1> sub ecx, ecx
3226 00004D31 B102 <1> mov cl, 2
3227 00004D33 E8E9D6FFFF <1> call WAITF ; NEC FLAGS UPDATE CYCLE
3228 <1> ; 09/08/2022
3229 <1> ; cf = 0, zf = 1
3230 <1> ;popf ; RESTORE FLAGS FOR EXIT
3231 <1> ;pop bx ; RESTORE REG
3232 00004D38 C3 <1> retn ; CY = 0 FROM TEST INSTRUCTION
3233 <1>
3234 <1> ;-----
3235 <1> ; SEEK
3236 <1> ; THIS ROUTINE WILL MOVE THE HEAD ON THE NAMED DRIVE TO THE NAMED
3237 <1> ; TRACK. IF THE DRIVE HAS NOT BEEN ACCESSED SINCE THE DRIVE
3238 <1> ; RESET COMMAND WAS ISSUED, THE DRIVE WILL BE RECALIBRATED.
3239 <1>
3240 <1> ; ON ENTRY: DI = DRIVE #
3241 <1> ; CH = TRACK #
3242 <1>
3243 <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
3244 <1> ; AX,BX,CX DX DESTROYED
3245 <1> ;-----
3246 <1> SEEK:
3247 00004D39 89FB <1> mov ebx, edi ; BX = DRIVE #
3248 00004D3B B001 <1> mov al, 1 ; ESTABLISH MASK FOR RECALIBRATE TEST
3249 <1> ; 06/08/2022
3250 <1> ;xchg cl, bl ; SET DRIVE VALUE INTO CL
3251 <1> ;rol al, cl ; SHIFT MASK BY THE DRIVE VALUE
3252 <1> ;xchg cl, bl ; RECOVER TRACK VALUE
3253 <1> ; 06/08/2022
3254 00004D3D 84C3 <1> test bl, al ; test bl, 1
3255 00004D3F 7402 <1> jz short seek_0
3256 00004D41 FEC0 <1> inc al ; shl al, 1
3257 <1> seek_0:
3258 00004D43 8405[25770100] <1> test al, [SEEK_STATUS] ; TEST FOR RECALIBRATE REQUIRED
3259 00004D49 7526 <1> jnz short J28A ; JUMP IF RECALIBRATE NOT REQUIRED
3260 <1>
3261 00004D4B 0805[25770100] <1> or [SEEK_STATUS], al ; TURN ON THE NO RECALIBRATE BIT IN FLAG
3262 00004D51 E862000000 <1> call RECAL ; RECALIBRATE DRIVE
3263 00004D56 730E <1> jnc short AFT_RECAL ; RECALIBRATE DONE
3264 <1>
3265 <1> ;----- ISSUE RECALIBRATE FOR 80 TRACK DISKETTES
3266 <1>
3267 00004D58 C605[28770100]00 <1> mov byte [DSKETTE_STATUS], 0 ; CLEAR OUT INVALID STATUS
3268 00004D5F E854000000 <1> call RECAL ; RECALIBRATE DRIVE
3269 00004D64 7251 <1> jc short RB ; IF RECALIBRATE FAILS TWICE THEN ERROR
3270 <1>
3271 <1> AFT_RECAL:
3272 00004D66 C687[35770100]00 <1> mov byte [DSK_TRK+edi], 0 ; SAVE NEW CYLINDER AS PRESENT POSITION
3273 00004D6D 08ED <1> or ch, ch ; CHECK FOR SEEK TO TRACK 0
3274 00004D6F 743F <1> jz short DO_WAIT ; HEAD SETTLE, CY = 0 IF JUMP
3275 <1>
3276 <1> ;----- DRIVE IS IN SYNCHRONIZATION WITH CONTROLLER, SEEK TO TRACK
3277 <1>
3278 00004D71 F687[33770100]20 <1> J28A: test byte [DSK_STATE+edi], DBL_STEP ; CHECK FOR DOUBLE STEP REQUIRED
3279 00004D78 7402 <1> jz short _R7 ; SINGLE STEP REQUIRED BYPASS DOUBLE
3280 00004D7A D0E5 <1> shl ch, 1 ; DOUBLE NUMBER OF STEP TO TAKE
3281 <1>
3282 00004D7C 3AAF[35770100] <1> _R7: cmp ch, [DSK_TRK+edi] ; SEE IF ALREADY AT THE DESIRED TRACK
3283 00004D82 7433 <1> je short RB ; IF YES, DO NOT NEED TO SEEK
3284 <1>
3285 00004D84 BA[B74D0000] <1> mov edx, NEC_ERR ; LOAD RETURN ADDRESS
3286 00004D89 52 <1> push edx ; (*) ; ON STACK FOR NEC OUTPUT ERROR
3287 00004D8A 88AF[35770100] <1> mov [DSK_TRK+edi], ch ; SAVE NEW CYLINDER AS PRESENT POSITION
3288 00004D90 B40F <1> mov ah, 0Fh ; SEEK COMMAND TO NEC
3289 00004D92 E86BFFFFFF <1> call NEC_OUTPUT
3290 00004D97 89FB <1> mov ebx, edi ; BX = DRIVE #
3291 00004D99 88DC <1> mov ah, bl ; OUTPUT DRIVE NUMBER
3292 00004D9B E862FFFFFF <1> call NEC_OUTPUT
3293 00004DA0 8AA7[35770100] <1> mov ah, [DSK_TRK+edi] ; GET CYLINDER NUMBER
3294 00004DA6 E857FFFFFF <1> call NEC_OUTPUT
3295 00004DAB E827000000 <1> call CHK_STAT_2 ; ENDING INTERRUPT AND SENSE STATUS
3296 <1>
3297 <1> ;----- WAIT FOR HEAD SETTLE
3298 <1>
3299 <1> DO_WAIT:
3300 00004DB0 9C <1> pushf ; SAVE STATUS
3301 00004DB1 E81AFFFFFF <1> call HD_WAIT ; WAIT FOR HEAD SETTLE TIME
3302 00004DB6 9D <1> popf ; RESTORE STATUS
3303 <1>
3304 <1> RB:
3305 <1> NEC_ERR:
3306 <1> ; 08/02/2015 (code trick here from original IBM PC/AT DISKETTE.ASM)
3307 00004DB7 C3 <1> ; (*) nec_err -> retn (push edx -> pop edx) -> nec_err -> retn
3308 <1> ; RETURN TO CALLER
3309 <1>
3310 <1> ;-----
3311 <1> ; RECAL
3312 <1> ; RECALIBRATE DRIVE
3313 <1>
3314 <1> ; ON ENTRY: DI = DRIVE #
3315 <1>
3316 <1> ; ON EXIT: CY REFLECTS STATUS OF OPERATION.
3317 <1> ;-----
3318 <1> RECAL:
3319 <1> ;push cx
3320 00004DB8 51 <1> ; 11/04/2021
3321 00004DB9 B8[D54D0000] <1> push ecx
3322 00004DBE 50 <1> mov eax, RC_BACK ; LOAD NEC_OUTPUT ERROR
3323 00004DBF B407 <1> push eax
3324 00004DC1 E83CFFFFFF <1> mov ah, 07h ; RECALIBRATE COMMAND
3325 00004DC6 89FB <1> call NEC_OUTPUT
3326 <1> mov ebx, edi ; BX = DRIVE #

```

```

3326 00004DC8 88DC      <1>      mov     ah, b1
3327 00004DCA E83FFFFFFF <1>      call    NEC_OUTPUT      ; OUTPUT THE DRIVE NUMBER
3328 00004DCF E803000000 <1>      call    CHK_STAT_2      ; GET THE INTERRUPT AND SENSE INT STATUS
3329 00004DD4 58        <1>      pop     eax            ; THROW AWAY ERROR
3330                                <1> RC_BACK:
3331                                <1>      ;pop     cx
3332                                <1>      ; 11/04/2021
3333 00004DD5 59        <1>      pop     ecx
3334 00004DD6 C3        <1>      retn
3335                                <1>
3336                                <1> -----
3337                                <1>      CHK_STAT_2
3338                                <1>      ; THIS ROUTINE HANDLES THE INTERRUPT RECEIVED AFTER RECALIBRATE,
3339                                <1>      ; OR SEEK TO THE ADAPTER. THE INTERRUPT IS WAITED FOR, THE
3340                                <1>      ; INTERRUPT STATUS SENSED, AND THE RESULT RETURNED TO THE CALLER.
3341                                <1>
3342                                <1>      ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
3343                                <1> -----
3344                                <1> CHK_STAT_2:
3345 00004DD7 B8[FF4D0000] <1>      mov     eax, CS_BACK      ; LOAD NEC_OUTPUT ERROR ADDRESS
3346 00004DDC 50        <1>      push    eax
3347 00004DDD E828000000 <1>      call    WAIT_INT          ; WAIT FOR THE INTERRUPT
3348 00004DE2 721A      <1>      jc      short J34        ; IF ERROR, RETURN IT
3349 00004DE4 B408      <1>      mov     ah, 08h          ; SENSE INTERRUPT STATUS COMMAND
3350 00004DE6 E817FFFFFF <1>      call    NEC_OUTPUT
3351 00004DEB E849000000 <1>      call    RESULTS          ; READ IN THE RESULTS
3352 00004DF0 720C      <1>      jc      short J34
3353 00004DF2 A0[29770100] <1>      mov     al, [NEC_STATUS]  ; GET THE FIRST STATUS BYTE
3354 00004DF7 2460      <1>      and     al, 01100000B    ; ISOLATE THE BITS
3355 00004DF9 3C60      <1>      cmp     al, 01100000B    ; TEST FOR CORRECT VALUE
3356 00004DFB 7403      <1>      jz      short J35        ; IF ERROR, GO MARK IT
3357 00004DFD F8        <1>      cld
3358                                <1>      J34:
3359 00004DFE 58        <1>      pop     eax            ; THROW AWAY ERROR RETURN
3360                                <1> CS_BACK:
3361 00004DFF C3        <1>      retn
3362                                <1> J35:
3363 00004E00 800D[28770100]40 <1>      or      byte [DSKETTE_STATUS], BAD_SEEK
3364 00004E07 F9        <1>      stc
3365 00004E08 EBF4      <1>      jmp     short J34        ; ERROR RETURN CODE
3366                                <1>
3367                                <1> -----
3368                                <1>      WAIT_INT
3369                                <1>      ; THIS ROUTINE WAITS FOR AN INTERRUPT TO OCCUR A TIME OUT ROUTINE
3370                                <1>      ; TAKES PLACE DURING THE WAIT, SO THAT AN ERROR MAY BE RETURNED
3371                                <1>      ; IF THE DRIVE IS NOT READY.
3372                                <1>
3373                                <1>      ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
3374                                <1> -----
3375                                <1>
3376                                <1>      ; 17/12/2014
3377                                <1>      ; 2.5 seconds waiting !
3378                                <1>      ; (AWARD BIOS - 1999, WAIT_FDU_INT_LOW, WAIT_FDU_INT_HI)
3379                                <1>      ; amount of time to wait for completion interrupt from NEC.
3380                                <1>
3381                                <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
3382                                <1>      WAIT_INT:
3383 00004E0A FB        <1>      sti
3384                                <1>      ; 06/08/2022
3385                                <1>      ; cld
3386                                <1>      ;
3387                                <1>      ; mov     b1, 10
3388                                <1>      ; xor     cx, cx
3389                                <1>      ;
3390                                <1>      ; Modification from AWARD BIOS - 1999 (ATORGS.ASM, WAIT
3391                                <1>      ;
3392                                <1>      ; WAIT_FOR_MEM:
3393                                <1>      ; waits for a bit at a specified memory location pointed
3394                                <1>      ; to by ES:[DI] to become set.
3395                                <1>      ; INPUT:
3396                                <1>      ; AH= Mask to test with.
3397                                <1>      ; ES:[DI] = memory location to watch.
3398                                <1>      ; BH: CX= Number of memory refresh periods to delay.
3399                                <1>      ; (normally 30 microseconds per period.)
3400                                <1>
3401                                <1>      ; waiting for (max.) 2.5 secs in 30 micro units.
3402                                <1>      ; mov     cx, WAIT_FDU_INT_LO      ; 017798
3403                                <1>      ; mov     b1, WAIT_FDU_INT_HI
3404                                <1>      ; mov     b1, WAIT_FDU_INT_HI + 1
3405                                <1>      ; 27/02/2015
3406 00004E0B B986450100 <1>      mov     ecx, WAIT_FDU_INT_LH ; 83334 (2.5 seconds)
3407                                <1>      WFMS_CHECK_MEM:
3408 00004E10 F605[25770100]80 <1>      test    byte [SEEK_STATUS], INT_FLAG
3409                                <1>      ; TEST FOR INTERRUPT OCCURRING
3410 00004E17 7516      <1>      jnz     short J37
3411                                <1>      WFMS_HI:
3412 00004E19 E461      <1>      in      al, PORT_B ; 061h ; SYS1, wait for lo to hi
3413 00004E1B A810      <1>      test    al, 010h ; transition on memory
3414 00004E1D 75FA      <1>      jnz     short WFMS_HI ; refresh.
3415                                <1>      WFMS_LO:
3416 00004E1F E461      <1>      in      al, PORT_B ; SYS1
3417 00004E21 A810      <1>      test    al, 010h
3418 00004E23 74FA      <1>      jz      short WFMS_LO
3419 00004E25 E2E9      <1>      loop    WFMS_CHECK_MEM
3420                                <1>
3421                                <1>      ; WFMS_OUTER_LP:
3422                                <1>      ; or      b1, b1
3423                                <1>      ; jz      short J36A
3424                                <1>      ; dec     b1
3425                                <1>      ; jz      short J36A
3426                                <1>      ; jmp     short WFMS_CHECK_MEM
3427                                <1>
3428                                <1>      ; 17/12/2014
3429                                <1>      ; 16/12/2014
3430                                <1>      ; mov     byte [wait_count], 0 ; Reset (INT 08H) counter
3431                                <1>      ; J36:
3432                                <1>      ; test    byte [SEEK_STATUS], INT_FLAG
3433                                <1>      ; TEST FOR INTERRUPT OCCURRING
3434                                <1>      ; jnz     short J37
3435                                <1>
3436                                <1>      ; 16/12/2014
3437                                <1>      ; loop    J36
3438                                <1>      ; dec     b1
3439                                <1>      ; jnz     short J36
3440                                <1>      ; cmp     byte [wait_count], 46 ; (46/18.2 seconds)
3441                                <1>      ; jb      short J36
3442                                <1>
3443                                <1>      ; WFMS_TIMEOUT:
3444                                <1>      ; J36A:
3445 00004E27 800D[28770100]80 <1>      or      byte [DSKETTE_STATUS], TIME_OUT ; NOTHING HAPPENED
3446 00004E2E F9        <1>      stc
3447                                <1>      ; ERROR RETURN
3448                                <1>      J37:
3449 00004E30 8025[25770100]7F <1>      pushf
3449 00004E30 8025[25770100]7F <1>      and     byte [SEEK_STATUS], ~INT_FLAG ; TURN OFF INTERRUPT FLAG

```

```

3450 00004E37 9D      <1>      popf                ; RECOVER CARRY
3451 00004E38 C3      <1>      retn                 ; GOOD RETURN CODE
3452                  <1>
3453                  <1> ;-----
3454                  <1> ; RESULTS
3455                  <1> ; THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER RETURNS
3456                  <1> ; FOLLOWING AN INTERRUPT.
3457                  <1>
3458                  <1> ; ON EXIT: @DSKETTE_STATUS, CY REFLECT STATUS OF OPERATION.
3459                  <1> ; AX,BX,CX,DX DESTROYED
3460                  <1> ;-----
3461                  <1> RESULTS:
3462                  <1> ; 06/08/2022 - TRDOS 386 v2.0.5
3463 00004E39 57      <1>      push     edi
3464 00004E3A BF[29770100] <1>      mov     edi, NEC_STATUS          ; POINTER TO DATA AREA
3465 00004E3F B307    <1>      mov     bl, 7                ; MAX STATUS BYTES
3466 00004E41 66BAF403 <1>      mov     dx, 03F4h             ; STATUS PORT
3467                  <1>
3468                  <1> ;----- WAIT FOR REQUEST FOR MASTER
3469                  <1>
3470                  <1> _R10:
3471                  <1> ; 16/12/2014
3472                  <1> ; wait for (max) 0.5 seconds
3473                  <1> ; mov     bh, 2                ; HIGH ORDER COUNTER
3474                  <1> ; xor     cx, cx                ; COUNTER
3475                  <1>
3476                  <1> ;Time to wait while waiting for each byte of NEC results = .5
3477                  <1> ;seconds. .5 seconds = 500,000 micros. 500,000/30 = 16,667.
3478                  <1> ; 27/02/2015
3479 00004E45 B91B410000 <1>      mov     ecx, WAIT_FDU_RESULTS_LH ; 16667
3480                  <1> ; mov     cx, WAIT_FDU_RESULTS_LO ; 16667
3481                  <1> ; mov     bh, WAIT_FDU_RESULTS_HI+1 ; 0+1
3482                  <1>
3483                  <1> WFPSR_OUTER_LP:
3484                  <1> ;
3485                  <1> WFPSR_CHECK_PORT:
3486                  <1> J39:
3487                  <1> ; in     al, dx                ; WAIT FOR MASTER
3488                  <1> ; and     al, 11000000b           ; GET STATUS
3489                  <1> ; cmp     al, 11000000b           ; KEEP ONLY STATUS AND DIRECTION
3490                  <1> ; jz      short J42             ; STATUS 1 AND DIRECTION 1 ?
3491                  <1> ; jz      short J42             ; STATUS AND DIRECTION OK
3492                  <1> WFPSR_HI:
3493                  <1> ; in     al, PORT_B          ; 061h ; SYS1 ; wait for hi to lo
3494                  <1> ; test    al, 010h           ; transition on memory
3495                  <1> ; jnz     short WFPSR_HI         ; refresh.
3496                  <1> WFPSR_LO:
3497                  <1> ; in     al, PORT_B          ; SYS1
3498                  <1> ; test    al, 010h           ;
3499                  <1> ; jz      SHORT WFPSR_LO         ;
3500                  <1> ; loop    WFPSR_CHECK_PORT
3501                  <1>
3502                  <1> ; ; 27/02/2015
3503                  <1> ; ; dec     bh
3504                  <1> ; ; jnz     short WFPSR_OUTER_LP
3505                  <1> ; ; jmp     short WFPSR_TIMEOUT ; fail
3506                  <1>
3507                  <1> ; ; mov     byte [wait_count], 0
3508                  <1> ; J39:
3509                  <1> ; ; in     al, dx                ; WAIT FOR MASTER
3510                  <1> ; ; and     al, 11000000b           ; GET STATUS
3511                  <1> ; ; cmp     al, 11000000b           ; KEEP ONLY STATUS AND DIRECTION
3512                  <1> ; ; jz      short J42             ; STATUS 1 AND DIRECTION 1 ?
3513                  <1> ; ; jz      short J42             ; STATUS AND DIRECTION OK
3514                  <1> ; ; loop    J39                ; LOOP TILL TIMEOUT
3515                  <1> ; ; dec     bh                ; DECREMENT HIGH ORDER COUNTER
3516                  <1> ; ; jnz     short J39                ; REPEAT TILL DELAY DONE
3517                  <1> ; ; cmp     byte [wait_count], 10 ; (10/18.2 seconds)
3518                  <1> ; ; jnb     short J39
3519                  <1>
3520 00004E5F 800D[28770100]80 <1> ; WFPSR_TIMEOUT:
3521 00004E66 F9      <1> ; or      byte [DSKETTE_STATUS], TIME_OUT
3522 00004E67 EB28    <1> ; stc
3523                  <1> ; jmp     short POPRES          ; SET ERROR RETURN
3524                  <1> ; ; POP REGISTERS AND RETURN
3525                  <1> ;----- READ IN THE STATUS
3526                  <1>
3527 00004E69 EB00    <1> ; J42:
3528                  <1> ; jmp     $+2                ; I/O DELAY
3529                  <1> ; ; inc     dx                ; POINT AT DATA PORT
3530                  <1> ; ; 06/08/2022
3531 00004E6B FEC2    <1> ; inc     dl
3532 00004E6D EC      <1> ; in     al, dx                ; GET THE DATA
3533                  <1> ; 16/12/2014
3534                  <1> ; NEWIODELAY
3535                  <2> out     0EBh, al
3536                  <1> ; mov     [edi], al          ; STORE THE BYTE
3537                  <1> ; inc     edi                ; INCREMENT THE POINTER
3538                  <1> ; ; 16/12/2014
3539                  <1> ; ; push    cx
3540                  <1> ; ; mov     cx, 30
3541                  <1> ; ; wdw2:
3542                  <1> ; ; NEWIODELAY
3543                  <1> ; ; loop    wdw2
3544                  <1> ; ; pop     cx
3545                  <1> ; ; mov     ecx, 3                ; MINIMUM 24 MICROSECONDS FOR NEC
3546                  <1> ; ; 11/04/2021
3547                  <1> ; ; mov     ecx, 2
3548                  <1> ; ; 06/08/2022
3549 00004E73 29C9    <1> ; sub     ecx, ecx
3550 00004E75 B102    <1> ; mov     cl, 2
3551 00004E77 E8A5D5FFFF <1> ; call    WAITF                ; WAIT 30 TO 45 MICROSECONDS
3552                  <1> ; dec     dx                ; POINT AT STATUS PORT
3553                  <1> ; ; 06/08/2022
3554 00004E7C FECA    <1> ; dec     dl
3555 00004E7E EC      <1> ; in     al, dx                ; GET STATUS
3556                  <1> ; 16/12/2014
3557                  <1> ; ; NEWIODELAY
3558                  <2> out     0EBh, al
3559                  <1> ;
3560 00004E81 A810    <1> ; test    al, 00010000b           ; TEST FOR NEC STILL BUSY
3561 00004E83 740C    <1> ; jz      short POPRES          ; RESULTS DONE ?
3562                  <1>
3563 00004E85 FECB    <1> ; dec     bl                ; DECREMENT THE STATUS COUNTER
3564 00004E87 75BC    <1> ; jnz     short _R10            ; GO BACK FOR MORE
3565 00004E89 800D[28770100]20 <1> ; or      byte [DSKETTE_STATUS], BAD_NEC ; TOO MANY STATUS BYTES
3566 00004E90 F9      <1> ; stc
3567                  <1> ; ; RESULT OPERATION IS DONE
3568                  <1> ; POPRES:
3569 00004E91 5F      <1> ; pop     edi
3570 00004E92 C3      <1> ; retn
3571                  <1> ; RETURN WITH CARRY SET

```

```

3572 <1> ;-----
3573 <1> READ_DSKCHNG
3574 <1> READS THE STATE OF THE DISK CHANGE LINE.
3575 <1>
3576 <1> ON ENTRY: DI = DRIVE #
3577 <1>
3578 <1> ON EXIT: DI = DRIVE #
3579 <1> ZF = 0 : DISK CHANGE LINE INACTIVE
3580 <1> ZF = 1 : DISK CHANGE LINE ACTIVE
3581 <1> AX,CX,DX DESTROYED
3582 <1> ;-----
3583 <1> READ_DSKCHNG:
3584 00004E93 E8AFFDFFFF <1> call MOTOR_ON ; TURN ON THE MOTOR IF OFF
3585 00004E98 66BAF703 <1> mov dx, 03F7h ; ADDRESS DIGITAL INPUT REGISTER
3586 00004E9C EC <1> in al, dx ; INPUT DIGITAL INPUT REGISTER
3587 00004E9D A880 <1> test al, DSK_CHG ; CHECK FOR DISK CHANGE LINE ACTIVE
3588 00004E9F C3 <1> retn ; RETURN TO CALLER WITH ZERO FLAG SET
3589 <1>
3590 <1> fdc_int:
3591 <1> ; 30/07/2015
3592 <1> ; 16/02/2015
3593 <1> ;int_0Eh: ; 11/12/2014
3594 <1>
3595 <1> ;--- HARDWARE INT 0EH -- ( IRQ LEVEL 6 ) -----
3596 <1> DISK_INT
3597 <1> THIS ROUTINE HANDLES THE DISKETTE INTERRUPT.
3598 <1>
3599 <1> ON EXIT: THE INTERRUPT FLAG IS SET IN @SEEK_STATUS.
3600 <1> ;-----
3601 <1> DISK_INT_1:
3602 <1> ;push ax ; SAVE WORK REGISTER
3603 <1> ; 11/04/2021
3604 00004EA0 50 <1> push eax
3605 00004EA1 1E <1> push ds
3606 00004EA2 66B81000 <1> mov ax, KDATA
3607 00004EA6 8ED8 <1> mov ds, ax
3608 00004EA8 800D[25770100]80 <1> or byte [SEEK_STATUS], INT_FLAG ; TURN ON INTERRUPT OCCURRED
3609 00004EAF B020 <1> mov al, EOI ; END OF INTERRUPT MARKER
3610 00004EB1 E620 <1> out INTA00, al ; INTERRUPT CONTROL PORT
3611 00004EB3 1F <1> pop ds
3612 <1> ;pop ax ; RECOVER REGISTER
3613 <1> ; 11/04/2021
3614 00004EB4 58 <1> pop eax
3615 00004EB5 CF <1> iretd ; RETURN FROM INTERRUPT
3616 <1>
3617 <1> ;-----
3618 <1> DSKETTE_SETUP
3619 <1> THIS ROUTINE DOES A PRELIMINARY CHECK TO SEE WHAT TYPE OF
3620 <1> DISKETTE DRIVES ARE ATTACH TO THE SYSTEM.
3621 <1> ;-----
3622 <1>
3623 <1> ; 09/08/2022 - TRDOS 386 kernel v2.0.5
3624 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
3625 <1>
3626 <1> DSKETTE_SETUP:
3627 <1> ;push ax ; SAVE REGISTERS
3628 <1> ;push bx
3629 <1> ;push cx
3630 00004EB6 52 <1> push edx
3631 <1> ;push di
3632 <1> ;push ds
3633 <1> ; 14/12/2014
3634 <1> ;mov word [DISK_POINTER], MD_TBL6
3635 <1> ;mov [DISK_POINTER+2], cs
3636 <1> ;
3637 <1> ;or byte [RTC_WAIT_FLAG], 1 ; NO RTC WAIT, FORCE USE OF LOOP
3638 00004EB7 31FF <1> xor edi, edi ; INITIALIZE DRIVE POINTER
3639 <1> ; 09/08/2022
3640 <1> ;mov esi, eax
3641 00004EB9 31C0 <1> xor eax, eax ; eax = 0
3642 00004EBB 66A3[33770100] <1> mov [DSK_STATE], ax ; INITIALIZE STATES
3643 00004EC1 8025[30770100]33 <1> and byte [LAstrate], ~(STRT_MSK+SEND_MSK) ; CLEAR START & SEND
3644 00004EC8 800D[30770100]C0 <1> or byte [LAstrate], SEND_MSK ; INITIALIZE SENT TO IMPOSSIBLE
3645 00004ECF A2[25770100] <1> mov [SEEK_STATUS], al ; INDICATE RECALIBRATE NEEDED
3646 00004ED4 A2[27770100] <1> mov [MOTOR_COUNT], al ; INITIALIZE MOTOR COUNT
3647 00004ED9 A2[26770100] <1> mov [MOTOR_STATUS], al ; INITIALIZE DRIVES TO OFF STATE
3648 00004EDE A2[28770100] <1> mov [DSKETTE_STATUS], al ; NO ERRORS
3649 <1> ;
3650 <1> ; 28/02/2015
3651 <1> ;mov word [cfd], 100h
3652 00004EE3 E815F4FFFF <1> call DSK_RESET
3653 00004EE8 5A <1> pop edx
3654 00004EE9 F8 <1> cld ; 29/05/2016
3655 00004EEA C3 <1> retn
3656 <1>
3657 <1> ;SUP0:
3658 <1> call DRIVE_DET ; DETERMINE DRIVE
3659 <1> call XLAT_OLD ; TRANSLATE STATE TO COMPATIBLE MODE
3660 <1> ; 02/01/2015
3661 <1> ;inc di ; POINT TO NEXT DRIVE
3662 <1> ;cmp di, MAX_DRV ; SEE IF DONE
3663 <1> ;jnz short SUP0 ; REPEAT FOR EACH ORIVE
3664 <1> ;cmp byte [fd1_type], 0
3665 <1> ;jna short sup1
3666 <1> ;or di, di
3667 <1> ;jnz short sup1
3668 <1> ;inc di
3669 <1> ;jmp short SUP0
3670 <1> ;sup1:
3671 <1> mov byte [SEEK_STATUS], 0 ; FORCE RECALIBRATE
3672 <1> ;and byte [RTC_WAIT_FLAG], 0FEh ; ALLOW FOR RTC WAIT
3673 <1> call SETUP_END ; VARIOUS CLEANUPS
3674 <1> ;pop ds ; RESTORE CALLERS REGISTERS
3675 <1> ;pop di
3676 <1> ;pop edx
3677 <1> ;pop cx
3678 <1> ;pop bx
3679 <1> ;pop ax
3680 <1> ;retn
3681 <1>
3682 <1> ;////////////////////////////////////
3683 <1> ; END OF DISKETTE I/O ;;;;;;;;;;;;;;;;;;;;;;;;;;
3684 <1>
3685 <1> ; 17/04/2021 (TRDOS 386 v2.0.4)
3686 <1>
3687 <1> ; 11/04/2021
3688 <1> ;int13h: ; 21/02/2015
3689 <1> ;pushfd
3690 <1> ;push cs
3691 <1> ;call DISK_IO
3692 <1> ;retn
3693 <1>
3694 <1> ;;;; DISK I/O ;;;;;;;;;;;;;;;;;;;;;;;;;; 21/02/2015 ;;;
3695 <1> ;////////////////////////////////////

```



```

3696 <1>
3697 <1> ; DISK I/O - Erdogan Tan (Retro UNIX 386 v1 project)
3698 <1> ; 18/02/2016
3699 <1> ; 17/02/2016
3700 <1> ; 23/02/2015
3701 <1> ; 21/02/2015 (unix386.s)
3702 <1> ; 22/12/2014 - 14/02/2015 (dsectrm2.s)
3703 <1>
3704 <1> ; Original Source Code:
3705 <1> ; DISK ---- 09/25/85 FIXED DISK BIOS
3706 <1> ; (IBM PC XT Model 286 System BIOS Source Code, 04-21-86)
3707 <1>
3708 <1> ; Modifications: by reference of AWARD BIOS 1999 (D1A0622)
3709 <1> ; Source Code - ATORGS.ASM, AHDSK.ASM
3710 <1>
3711 <1>
3712 <1> ;The wait for controller to be not busy is 10 seconds.
3713 <1> ;10,000,000 / 30 = 333,333. 333,333 decimal = 051615h
3714 <1> ;;WAIT_HDU_CTLR_BUSY_LO equ 1615h
3715 <1> ;;WAIT_HDU_CTLR_BUSY_HI equ 05h
3716 <1> WAIT_HDU_CTLR_BUSY_LH equ 51615h ; 21/02/2015
3717 <1>
3718 <1> ;The wait for controller to issue completion interrupt is 10 seconds.
3719 <1> ;10,000,000 / 30 = 333,333. 333,333 decimal = 051615h
3720 <1> ;;WAIT_HDU_INT_LO equ 1615h
3721 <1> ;;WAIT_HDU_INT_HI equ 05h
3722 <1> WAIT_HDU_INT_LH equ 51615h ; 21/02/2015
3723 <1>
3724 <1> ;The wait for Data request on read and write longs is
3725 <1> ;2000 us. (?)
3726 <1> ;;WAIT_HDU_DRQ_LO equ 1000 ; 03E8h
3727 <1> ;;WAIT_HDU_DRQ_HI equ 0
3728 <1> WAIT_HDU_DRQ_LH equ 1000 ; 21/02/2015
3729 <1>
3730 <1> ; Port 61h (PORT_B)
3731 <1> SYS1equ 61h ; PORT_B (diskette.inc)
3732 <1>
3733 <1> ; 23/12/2014
3734 <1> %define CMD_BLOCK ebp-8 ; 21/02/2015
3735 <1>
3736 <1> ;--- INT 13H -----
3737 <1>
3738 <1> ; FIXED DISK I/O INTERFACE
3739 <1>
3740 <1> ; THIS INTERFACE PROVIDES ACCESS TO 5 1/4" FIXED DISKS THROUGH :
3741 <1> ; THE IBM FIXED DISK CONTROLLER. :
3742 <1>
3743 <1> ; THE BIOS ROUTINES ARE MEANT TO BE ACCESSED THROUGH :
3744 <1> ; SOFTWARE INTERRUPTS ONLY. ANY ADDRESSES PRESENT IN :
3745 <1> ; THESE LISTINGS ARE INCLUDED ONLY FOR COMPLETENESS, :
3746 <1> ; NOT FOR REFERENCE. APPLICATIONS WHICH REFERENCE ANY :
3747 <1> ; ABSOLUTE ADDRESSES WITHIN THE CODE SEGMENTS OF BIOS :
3748 <1> ; VIOLATE THE STRUCTURE AND DESIGN OF BIOS. :
3749 <1>
3750 <1> ;-----
3751 <1>
3752 <1> ; INPUT (AH)= HEX COMMAND VALUE
3753 <1>
3754 <1> ; (AH)= 00H RESET DISK (DL = 80H,81H) / DISKETTE
3755 <1> ; (AH)= 01H READ THE STATUS OF THE LAST DISK OPERATION INTO (AL) :
3756 <1> ; NOTE: DL < 80H - DISKETTE :
3757 <1> ; DL > 80H - DISK :
3758 <1> ; (AH)= 02H READ THE DESIRED SECTORS INTO MEMORY
3759 <1> ; (AH)= 03H WRITE THE DESIRED SECTORS FROM MEMORY
3760 <1> ; (AH)= 04H VERIFY THE DESIRED SECTORS :
3761 <1> ; (AH)= 05H FORMAT THE DESIRED TRACK :
3762 <1> ; (AH)= 06H UNUSED :
3763 <1> ; (AH)= 07H UNUSED :
3764 <1> ; (AH)= 08H RETURN THE CURRENT DRIVE PARAMETERS :
3765 <1> ; (AH)= 09H INITIALIZE DRIVE PAIR CHARACTERISTICS :
3766 <1> ; INTERRUPT 41 POINTS TO DATA BLOCK FOR DRIVE 0 :
3767 <1> ; INTERRUPT 46 POINTS TO DATA BLOCK FOR DRIVE 1 :
3768 <1> ; (AH)= 0AH READ LONG
3769 <1> ; (AH)= 0BH WRITE LONG (READ & WRITE LONG ENCOMPASS 512 + 4 BYTES ECC) :
3770 <1> ; (AH)= 0CH SEEK
3771 <1> ; (AH)= 0DH ALTERNATE DISK RESET (SEE DL)
3772 <1> ; (AH)= 0EH UNUSED
3773 <1> ; (AH)= 0FH UNUSED
3774 <1> ; (AH)= 10H TEST DRIVE READY
3775 <1> ; (AH)= 11H RECALIBRATE
3776 <1> ; (AH)= 12H UNUSED
3777 <1> ; (AH)= 13H UNUSED
3778 <1> ; (AH)= 14H CONTROLLER INTERNAL DIAGNOSTIC
3779 <1> ; (AH)= 15H READ DASD TYPE
3780 <1>
3781 <1> ;-----
3782 <1>
3783 <1> ; REGISTERS USED FOR FIXED DISK OPERATIONS
3784 <1>
3785 <1> ; (DL) - DRIVE NUMBER (80H-81H FOR DISK. VALUE CHECKED) :
3786 <1> ; (DH) - HEAD NUMBER (0-15 ALLOWED, NOT VALUE CHECKED) :
3787 <1> ; (CH) - CYLINDER NUMBER (0-1023, NOT VALUE CHECKED)(SEE CL):
3788 <1> ; (CL) - SECTOR NUMBER (1-17, NOT VALUE CHECKED) :
3789 <1>
3790 <1> ; NOTE: HIGH 2 BITS OF CYLINDER NUMBER ARE PLACED :
3791 <1> ; IN THE HIGH 2 BITS OF THE CL REGISTER :
3792 <1> ; (10 BITS TOTAL) :
3793 <1>
3794 <1> ; (AL) - NUMBER OF SECTORS (MAXIMUM POSSIBLE RANGE 1-80H, :
3795 <1> ; FOR READ/WRITE LONG 1-79H) :
3796 <1>
3797 <1> ; (ES:BX) - ADDRESS OF BUFFER FOR READS AND WRITES, :
3798 <1> ; (NOT REQUIRED FOR VERIFY) :
3799 <1>
3800 <1> ; FORMAT (AH=5) ES:BX POINTS TO A 512 BYTE BUFFER. THE FIRST :
3801 <1> ; 2*(SECTORS/TRACK) BYTES CONTAIN F,N FOR EACH SECTOR.:
3802 <1> ; F = 00H FOR A GOOD SECTOR
3803 <1> ; 80H FOR A BAD SECTOR
3804 <1> ; N = SECTOR NUMBER
3805 <1> ; FOR AN INTERLEAVE OF 2 AND 17 SECTORS/TRACK :
3806 <1> ; THE TABLE SHOULD BE:
3807 <1>
3808 <1> ; DB 00H,01H,00H,0AH,00H,02H,00H,0BH,00H,03H,00H,0CH :
3809 <1> ; DB 00H,04H,00H,0DH,00H,05H,00H,0EH,00H,06H,00H,0FH :
3810 <1> ; DB 00H,07H,00H,10H,00H,08H,00H,11H,00H,09H :
3811 <1>
3812 <1> ;-----
3813 <1>
3814 <1> ;-----
3815 <1> ; OUTPUT
3816 <1> ; AH = STATUS OF CURRENT OPERATION
3817 <1> ; STATUS BITS ARE DEFINED IN THE EQUATES BELOW
3818 <1> ; CY = 0 SUCCESSFUL OPERATION (AH=0 ON RETURN) :
3819 <1> ; CY = 1 FAILED OPERATION (AH HAS ERROR REASON) :

```

```

3820 <1> ;
3821 <1> ; NOTE: ERROR 11H INDICATES THAT THE DATA READ HAD A RECOVERABLE :
3822 <1> ; ERROR WHICH WAS CORRECTED BY THE ECC ALGORITHM. THE DATA :
3823 <1> ; IS PROBABLY GOOD, HOWEVER THE BIOS ROUTINE INDICATES AN :
3824 <1> ; ERROR TO ALLOW THE CONTROLLING PROGRAM A CHANCE TO DECIDE :
3825 <1> ; FOR ITSELF. THE ERROR MAY NOT RECUR IF THE DATA IS :
3826 <1> ; REWRITTEN. :
3827 <1> ; :
3828 <1> ; IF DRIVE PARAMETERS WERE REQUESTED (DL >= 80H), :
3829 <1> ; INPUT: :
3830 <1> ; (DL) = DRIVE NUMBER :
3831 <1> ; ; 27/05/2016 - TRDOS 386 (TRDOS v2.0) :
3832 <1> ; EBX = Buffer address for fixed disk parameters table (32 bytes) :
3833 <1> ; OUTPUT: :
3834 <1> ; (DL) = NUMBER OF CONSECUTIVE ACKNOWLEDGING DRIVES ATTACHED (1-2) :
3835 <1> ; (CONTROLLER CARD ZERO TALLY ONLY) :
3836 <1> ; (DH) = MAXIMUM USEABLE VALUE FOR HEAD NUMBER :
3837 <1> ; (CH) = MAXIMUM USEABLE VALUE FOR CYLINDER NUMBER :
3838 <1> ; (CL) = MAXIMUM USEABLE VALUE FOR SECTOR NUMBER :
3839 <1> ; AND CYLINDER NUMBER HIGH BITS :
3840 <1> ; :
3841 <1> ; IF READ DASD TYPE WAS REQUESTED, :
3842 <1> ; :
3843 <1> ; AH = 0 - NOT PRESENT :
3844 <1> ; 1 - DISKETTE - NO CHANGE LINE AVAILABLE :
3845 <1> ; 2 - DISKETTE - CHANGE LINE AVAILABLE :
3846 <1> ; 3 - FIXED DISK :
3847 <1> ; :
3848 <1> ; CX,DX = NUMBER OF 512 BYTE BLOCKS WHEN AH = 3 :
3849 <1> ; :
3850 <1> ; REGISTERS WILL BE PRESERVED EXCEPT WHEN THEY ARE USED TO RETURN :
3851 <1> ; INFORMATION. :
3852 <1> ; :
3853 <1> ; NOTE: IF AN ERROR IS REPORTED BY THE DISK CODE, THE APPROPRIATE :
3854 <1> ; ACTION IS TO RESET THE DISK, THEN RETRY THE OPERATION. :
3855 <1> ; :
3856 <1> ; -----
3857 <1> ;
3858 <1> SENSE_FAIL EQU 0FFH ; NOT IMPLEMENTED
3859 <1> NO_ERR EQU 0E0H ; STATUS ERROR/ERROR REGISTER=0
3860 <1> WRITE_FAULT EQU 0CCH ; WRITE FAULT ON SELECTED DRIVE
3861 <1> UNDEF_ERR EQU 0BBH ; UNDEFINED ERROR OCCURRED
3862 <1> NOT_RDY EQU 0AAH ; DRIVE NOT READY
3863 <1> TIME_OUT EQU 80H ; ATTACHMENT FAILED TO RESPOND
3864 <1> BAD_SEEK EQU 40H ; SEEK OPERATION FAILED
3865 <1> BAD_CNTLRL EQU 20H ; CONTROLLER HAS FAILED
3866 <1> DATA_CORRECTED EQU 11H ; ECC CORRECTED DATA ERROR
3867 <1> BAD_ECC EQU 10H ; BAD ECC ON DISK READ
3868 <1> BAD_TRACK EQU 0BH ; NOT IMPLEMENTED
3869 <1> BAD_SECTOR EQU 0AH ; BAD SECTOR FLAG DETECTED
3870 <1> ;DMA_BOUNDARY EQU 09H ; DATA EXTENDS TOO FAR
3871 <1> INIT_FAIL EQU 07H ; DRIVE PARAMETER ACTIVITY FAILED
3872 <1> BAD_RESET EQU 05H ; RESET FAILED
3873 <1> ;RECORD_NOT_FND EQU 04H ; REQUESTED SECTOR NOT FOUND
3874 <1> ;BAD_ADDR_MARK EQU 02H ; ADDRESS MARK NOT FOUND
3875 <1> ;BAD_CMD EQU 01H ; BAD COMMAND PASSED TO DISK I/O
3876 <1> ;
3877 <1> ; -----
3878 <1> ;
3879 <1> ; FIXED DISK PARAMETER TABLE :
3880 <1> ; - THE TABLE IS COMPOSED OF A BLOCK DEFINED AS: :
3881 <1> ; :
3882 <1> ; +0 (1 WORD) - MAXIMUM NUMBER OF CYLINDERS :
3883 <1> ; +2 (1 BYTE) - MAXIMUM NUMBER OF HEADS :
3884 <1> ; +3 (1 WORD) - NOT USED/SEE PC-XT :
3885 <1> ; +5 (1 WORD) - STARTING WRITE PRECOMPENSATION CYL :
3886 <1> ; +7 (1 BYTE) - MAXIMUM ECC DATA BURST LENGTH :
3887 <1> ; +8 (1 BYTE) - CONTROL BYTE :
3888 <1> ; BIT 7 DISABLE RETRIES -OR- :
3889 <1> ; BIT 6 DISABLE RETRIES :
3890 <1> ; BIT 3 MORE THAN 8 HEADS :
3891 <1> ; +9 (3 BYTES) - NOT USED/SEE PC-XT :
3892 <1> ; +12 (1 WORD) - LANDING ZONE :
3893 <1> ; +14 (1 BYTE) - NUMBER OF SECTORS/TRACK :
3894 <1> ; +15 (1 BYTE) - RESERVED FOR FUTURE USE :
3895 <1> ; :
3896 <1> ; - TO DYNAMICALLY DEFINE A SET OF PARAMETERS: :
3897 <1> ; BUILD A TABLE FOR UP TO 15 TYPES AND PLACE :
3898 <1> ; THE CORRESPONDING VECTOR INTO INTERRUPT 41 :
3899 <1> ; FOR DRIVE 0 AND INTERRUPT 46 FOR DRIVE 1.: :
3900 <1> ; :
3901 <1> ; -----
3902 <1> ;
3903 <1> ;
3904 <1> ;
3905 <1> ; HARDWARE SPECIFIC VALUES :
3906 <1> ; :
3907 <1> ; - CONTROLLER I/O PORT :
3908 <1> ; :
3909 <1> ; > WHEN READ FROM: :
3910 <1> ; HF_PORT+0 - READ DATA (FROM CONTROLLER TO CPU) :
3911 <1> ; HF_PORT+1 - GET ERROR REGISTER :
3912 <1> ; HF_PORT+2 - GET SECTOR COUNT :
3913 <1> ; HF_PORT+3 - GET SECTOR NUMBER :
3914 <1> ; HF_PORT+4 - GET CYLINDER LOW :
3915 <1> ; HF_PORT+5 - GET CYLINDER HIGH (2 BITS) :
3916 <1> ; HF_PORT+6 - GET SIZE/DRIVE/HEAD :
3917 <1> ; HF_PORT+7 - GET STATUS REGISTER :
3918 <1> ; :
3919 <1> ; > WHEN WRITTEN TO: :
3920 <1> ; HF_PORT+0 - WRITE DATA (FROM CPU TO CONTROLLER) :
3921 <1> ; HF_PORT+1 - SET PRECOMPENSATION CYLINDER :
3922 <1> ; HF_PORT+2 - SET SECTOR COUNT :
3923 <1> ; HF_PORT+3 - SET SECTOR NUMBER :
3924 <1> ; HF_PORT+4 - SET CYLINDER LOW :
3925 <1> ; HF_PORT+5 - SET CYLINDER HIGH (2 BITS) :
3926 <1> ; HF_PORT+6 - SET SIZE/DRIVE/HEAD :
3927 <1> ; HF_PORT+7 - SET COMMAND REGISTER :
3928 <1> ; :
3929 <1> ; -----
3930 <1> ;
3931 <1> ; HF_PORT EQU 01F0H ; DISK PORT
3932 <1> ; HF1_PORT equ 0170h
3933 <1> ; HF_REG_PORT EQU 03F6H
3934 <1> ; HF1_REG_PORT equ 0376h
3935 <1> ;
3936 <1> ; HDC1_BASEPORT equ 1F0h
3937 <1> ; HDC2_BASEPORT equ 170h
3938 <1> ;
3939 <1> align 2
3940 <1> ;
3941 <1> ;----- STATUS REGISTER
3942 <1> ;
3943 <1> ST_ERROR EQU 0000001B ;

```

00004EEB 90

```

3944 <1> ST_INDEX EQU 00000010B ;
3945 <1> ST_CORRCTD EQU 00000100B ; ECC CORRECTION SUCCESSFUL
3946 <1> ST_DRQ EQU 00001000B ;
3947 <1> ST_SEEK_COMPL EQU 00010000B ; SEEK COMPLETE
3948 <1> ST_WRT_FLT EQU 00100000B ; WRITE FAULT
3949 <1> ST_READY EQU 01000000B ;
3950 <1> ST_BUSY EQU 10000000B ;
3951 <1>
3952 <1> ;----- ERROR REGISTER
3953 <1>
3954 <1> ERR_DAM EQU 00000001B ; DATA ADDRESS MARK NOT FOUND
3955 <1> ERR_TRK_0 EQU 00000010B ; TRACK 0 NOT FOUND ON RECAL
3956 <1> ERR_ABORT EQU 00000100B ; ABORTED COMMAND
3957 <1> ; EQU 00001000B ; NOT USED
3958 <1> ERR_ID EQU 00010000B ; ID NOT FOUND
3959 <1> ; EQU 00100000B ; NOT USED
3960 <1> ERR_DATA_ECC EQU 01000000B
3961 <1> ERR_BAD_BLOCK EQU 10000000B
3962 <1>
3963 <1>
3964 <1> RECAL_CMD EQU 00010000B ; DRIVE RECAL (10H)
3965 <1> READ_CMD EQU 00100000B ; READ (20H)
3966 <1> WRITE_CMD EQU 00110000B ; WRITE (30H)
3967 <1> VERIFY_CMD EQU 01000000B ; VERIFY (40H)
3968 <1> FMTTRK_CMD EQU 01010000B ; FORMAT TRACK (50H)
3969 <1> INIT_CMD EQU 01100000B ; INITIALIZE (60H)
3970 <1> SEEK_CMD EQU 01110000B ; SEEK (70H)
3971 <1> DIAG_CMD EQU 10010000B ; DIAGNOSTIC (90H)
3972 <1> SET_PARM_CMD EQU 10010001B ; DRIVE PARMS (91H)
3973 <1> NO_RETRIES EQU 00000001B ; CHD MODIFIER (01H)
3974 <1> ECC_MODE EQU 00000010B ; CMD MODIFIER (02H)
3975 <1> BUFFER_MODE EQU 00001000B ; CMD MODIFIER (08H)
3976 <1>
3977 <1> ;MAX_FILE EQU 2
3978 <1> ;S_MAX_FILE EQU 2
3979 <1> MAX_FILE equ 4 ; 22/12/2014
3980 <1> S_MAX_FILE equ 4 ; 22/12/2014
3981 <1>
3982 <1> DELAY_1 EQU 25H ; DELAY FOR OPERATION COMPLETE
3983 <1> DELAY_2 EQU 0600H ; DELAY FOR READY
3984 <1> DELAY_3 EQU 0100H ; DELAY FOR DATA REQUEST
3985 <1>
3986 <1> HF_FAIL EQU 08H ; CMOS FLAG IN BYTE 0EH
3987 <1>
3988 <1> ;----- COMMAND BLOCK REFERENCE
3989 <1>
3990 <1> ;CMD_BLOCK EQU BP-8 ; @CMD_BLOCK REFERENCES BLOCK HEAD IN SS
3991 <1> ; (BP) POINTS TO COMMAND BLOCK TAIL
3992 <1> ; AS DEFINED BY THE "ENTER" PARMS
3993 <1> ; 19/12/2014
3994 <1> ORG_VECTOR equ 4*13h ; INT 13h vector
3995 <1> DISK_VECTOR equ 4*40h ; INT 40h vector (for floppy disks)
3996 <1> ;HDISK_INT equ 4*76h ; Primary HDC - Hardware interrupt (IRQ14)
3997 <1> ;HDISK_INT1 equ 4*76h ; Primary HDC - Hardware interrupt (IRQ14)
3998 <1> ;HDISK_INT2 equ 4*77h ; Secondary HDC - Hardware interrupt (IRQ15)
3999 <1> ;HF_TBL_VEC equ 4*41h ; Pointer to 1st fixed disk parameter table
4000 <1> ;HF1_TBL_VEC equ 4*46h ; Pointer to 2nd fixed disk parameter table
4001 <1>
4002 <1> align 2
4003 <1>
4004 <1> ;-----
4005 <1> ; FIXED DISK I/O SETUP :
4006 <1> ; :
4007 <1> ; - ESTABLISH TRANSFER VECTORS FOR THE FIXED DISK :
4008 <1> ; - PERFORM POWER ON DIAGNOSTICS :
4009 <1> ; SHOULD AN ERROR OCCUR A "1701" MESSAGE IS DISPLAYED :
4010 <1> ; :
4011 <1> ;-----
4012 <1>
4013 <1> ; 09/08/2022
4014 <1> ; 06/08/2022 - TRDOS 386 Kernel v2.0.5
4015 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
4016 <1>
4017 <1> DISK_SETUP:
4018 <1> ;CLI
4019 <1> ;mov ax, ABS0 ; GET ABSOLUTE SEGMENT
4020 <1> ;xor ax, ax
4021 <1> ;mov ds, ax ; SET SEGMENT REGISTER
4022 <1> ;mov ax, [ORG_VECTOR] ; GET DISKETTE VECTOR
4023 <1> ;mov [DISK_VECTOR], ax ; INTO INT 40H
4024 <1> ;mov ax, [ORG_VECTOR+2]
4025 <1> ;mov [DISK_VECTOR+2], ax
4026 <1> ;mov word [ORG_VECTOR], DISK_IO ; FIXED DISK HANDLER
4027 <1> ;mov [ORG_VECTOR+2], cs
4028 <1> ; 1st controller (primary master, slave) - IRQ 14
4029 <1> ;mov word [HDISK_INT], HD_INT ; FIXED DISK INTERRUPT
4030 <1> ;mov word [HDISK_INT1], HD_INT ;
4031 <1> ;mov [HDISK_INT+2], cs
4032 <1> ;mov [HDISK_INT1+2], cs
4033 <1> ; 2nd controller (secondary master, slave) - IRQ 15
4034 <1> ;mov word [HDISK_INT2], HD1_INT ;
4035 <1> ;mov [HDISK_INT2+2], cs
4036 <1> ;
4037 <1> ;mov word [HF_TBL_VEC], HD0_DPT ; PARM TABLE DRIVE 80
4038 <1> ;mov word [HF_TBL_VEC+2], DPT_SEGM
4039 <1> ;mov word [HF1_TBL_VEC], HD1_DPT ; PARM TABLE DRIVE 81
4040 <1> ;mov word [HF1_TBL_VEC+2], DPT_SEGM
4041 <1> ;push cs
4042 <1> ;pop ds
4043 <1> ;mov word [HDPM_TBL_VEC], HD0_DPT ; PARM TABLE DRIVE 80h
4044 <1> ;mov word [HDPM_TBL_VEC+2], DPT_SEGM
4045 00004EEC C705[3C770100]0000- <1> mov dword [HDPM_TBL_VEC], (DPT_SEGM*16)+HD0_DPT
4045 00004EF4 0900 <1>
4046 <1> ;mov word [HDPS_TBL_VEC], HD1_DPT ; PARM TABLE DRIVE 81h
4047 <1> ;mov word [HDPS_TBL_VEC+2], DPT_SEGM
4048 00004EF6 C705[40770100]2000- <1> mov dword [HDPS_TBL_VEC], (DPT_SEGM*16)+HD1_DPT
4048 00004EFE 0900 <1>
4049 <1> ;mov word [HDSM_TBL_VEC], HD2_DPT ; PARM TABLE DRIVE 82h
4050 <1> ;mov word [HDSM_TBL_VEC+2], DPT_SEGM
4051 00004F00 C705[44770100]4000- <1> mov dword [HDSM_TBL_VEC], (DPT_SEGM*16)+HD2_DPT
4051 00004F08 0900 <1>
4052 <1> ;mov word [HDSS_TBL_VEC], HD3_DPT ; PARM TABLE DRIVE 83h
4053 <1> ;mov word [HDSS_TBL_VEC+2], DPT_SEGM
4054 00004F0A C705[48770100]6000- <1> mov dword [HDSS_TBL_VEC], (DPT_SEGM*16)+HD3_DPT
4054 00004F12 0900 <1>
4055 <1> ;
4056 <1> ;in al, INTB01 ; TURN ON SECOND INTERRUPT CHIP
4057 <1> ;and al, 0BFh
4058 <1> ;and al, 3Fh ; enable IRQ 14 and IRQ 15
4059 <1> ;jmp $+2
4060 <1> ;IODELAY
4061 <1> ;out INTB01, al
4062 <1> ;IODELAY
4063 <1> ;in al, INTA01 ; LET INTERRUPTS PASS THRU TO

```

```

4064      <1>      ;;and al, 0FBh          ; SECOND CHIP
4065      <1>      ;;jmp $+2
4066      <1>      ;;IODELAY
4067      <1>      ;;out INTA01, al
4068      <1>
4069      <1>      ;;sti
4070      <1>      ;;push ds              ; MOVE ABS0 POINTER TO
4071      <1>      ;;pop es               ; EXTRA SEGMENT POINTER
4072      <1>      ;;call DDS            ; ESTABLISH DATA SEGMENT
4073      <1>      ;;mov byte [DISK_STATUS1], 0      ; RESET THE STATUS INDICATOR
4074      <1>      ;;mov byte [HF_NUM], 0           ; ZERO NUMBER OF FIXED DISKS
4075      <1>      ;;mov byte [CONTROL_BYTE], 0
4076      <1>      ;;mov byte [PORT_OFF], 0         ; ZERO CARD OFFSET
4077      <1>      ; 20/12/2014 - private code by Erdogan Tan
4078      <1>      ; (out of original PC-AT, PC-XT BIOS code)
4079      <1>      ;mov si, hd0_type
4080      <1>      mov esi, hd0_type
4081      <1>      ;;mov cx, 4
4082      <1>      ;mov ecx, 4
4083      <1>      ; 06/08/2022
4084      <1>      sub ecx, ecx
4085      <1>      mov cl, 4
4086      <1>      hde_l:
4087      <1>      lodsb
4088      <1>      cmp al, 80h          ; 8?h = existing
4089      <1>      jnb short _L4
4090      <1>      inc byte [HF_NUM]    ; + 1 hard (fixed) disk drives
4091      <1>      _L4: ; 26/02/2015
4092      <1>      loop hde_l
4093      <1>      ;_L4: ; 0 <= [HF_NUM] =< 4
4094      <1>      ;_L4:
4095      <1>      ;; 31/12/2014 - cancel controller diagnostics here
4096      <1>      ;;mov cx, 3 ; 26/12/2014 (Award BIOS 1999)
4097      <1>      ;;mov cl, 3
4098      <1>
4099      <1>      ;;mov DL, 80H          ; CHECK THE CONTROLLER
4100      <1>      ;;hdc_d1:
4101      <1>      ;;mov AH, 14H          ; USE CONTROLLER DIAGNOSTIC COMMAND
4102      <1>      ;;INT 13H              ; CALL BIOS WITH DIAGNOSTIC COMMAND
4103      <1>      ;;jc short CTL_ERRX    ; DISPLAY ERROR MESSAGE IF BAD RETURN
4104      <1>      ;;jc short POD_DONE ;22/12/2014
4105      <1>      ;;jnc short hdc_reset0
4106      <1>      ;;loop hdc_d1
4107      <1>      ;; 27/12/2014
4108      <1>      ;;stc
4109      <1>      ;;retn
4110      <1>
4111      <1>      ;;hdc_reset0:
4112      <1>      ; 18/01/2015
4113      <1>      mov cl, [HF_NUM]
4114      <1>      and cl, cl
4115      <1>      jz short POD_DONE
4116      <1>
4117      <1>      mov dl, 7Fh
4118      <1>      hdc_reset1:
4119      <1>      inc dl
4120      <1>      ; 31/12/2015
4121      <1>      ;;push dx
4122      <1>      ;;push cx
4123      <1>      ;;push ds
4124      <1>      ;;sub ax, ax
4125      <1>      ;;mov ds, ax
4126      <1>      ;;mov ax, [TIMER_LOW]    ; GET START TIMER COUNTS
4127      <1>      ;;pop ds
4128      <1>      ;;mov bx, ax
4129      <1>      ;;add ax, 6*182          ; 60 SECONDS* 18.2
4130      <1>      ;;mov cx, ax
4131      <1>      ;;mov word [wait_count], 0 ; 22/12/2014 (reset wait counter)
4132      <1>
4133      <1>      ; 31/12/2014 - cancel HD_RESET_1
4134      <1>      ;;call HD_RESET_1        ; SET UP DRIVE 0, (1,2,3)
4135      <1>      ;;pop cx
4136      <1>      ;;pop dx
4137      <1>
4138      <1>      ; 18/01/2015
4139      <1>      mov ah, 0Dh ; ALTERNATE RESET
4140      <1>      ;int 13h
4141      <1>      call int13h
4142      <1>      loop hdc_reset1
4143      <1>      ;clc ; 29/05/2016
4144      <1>      POD_DONE:
4145      <1>      retn
4146      <1>
4147      <1>      ;;----- POD_ERROR
4148      <1>
4149      <1>      ;;CTL_ERRX:
4150      <1>      ;mov SI, OFFSET F1782      ; CONTROLLER ERROR
4151      <1>      ;call SET_FAIL              ; DO NOT IPL FROM DISK
4152      <1>      ;call E_MSG                 ; DISPLAY ERROR AND SET (BP) ERROR FLAG
4153      <1>      ;jmp short POD_DONE
4154      <1>
4155      <1>      ;;HD_RESET_1:
4156      <1>      ;push BX                  ; SAVE TIMER LIMITS
4157      <1>      ;push CX
4158      <1>      ;;RES_1: mov AH, 09H          ; SET DRIVE PARAMETERS
4159      <1>      ;INT 13H
4160      <1>      ;jc short RES_2
4161      <1>      ;mov AH, 11h                ; RECALIBRATE DRIVE
4162      <1>      ;INT 13H
4163      <1>      ;jnc short RES_CHK          ; DRIVE OK
4164      <1>      ;;RES_2: call POD_TCHK        ; CHECK TIME OUT
4165      <1>      ;cmp word [wait_count], 6*182 ; waiting time (in timer ticks)
4166      <1>      ; ; (30 seconds)
4167      <1>      ;cmc
4168      <1>      ;jnc short RES_1
4169      <1>      ;jb short RES_1
4170      <1>      ;;RES_FL: mov SI, OFFSET F1781 ; INDICATE DISK 1 FAILURE;
4171      <1>      ;test DL, 1
4172      <1>      ;jnz short RES_E1
4173      <1>      ;mov SI, OFFSET F1780      ; INDICATE DISK 0 FAILURE
4174      <1>      ;call SET_FAIL              ; DO NOT TRY TO IPL DISK 0
4175      <1>      ;jmp SHORT RES_E1
4176      <1>      ;;RES_ER: ; 22/12/2014
4177      <1>      ;RES_OK:
4178      <1>      ;pop CX                    ; RESTORE TIMER LIMITS
4179      <1>      ;pop BX
4180      <1>      ;retn
4181      <1>
4182      <1>      ;;RES_RS: mov AH, 00H          ; RESET THE DRIVE
4183      <1>      ;INT 13H
4184      <1>      ;;RES_CK: mov AH, 08H          ; GET MAX CYLINDER, HEAD, SECTOR
4185      <1>      ;mov BL, DL                ; SAVE DRIVE CODE
4186      <1>      ;INT 13H
4187      <1>      ;jc short RES_ER

```

```

4188 <1> ;; mov [NEC_STATUS],CX ; SAVE MAX CYLINDER, SECTOR
4189 <1> ;; mov DL,BL ; RESTORE DRIVE CODE
4190 <1> ;; RES_3: mov AX,0401H ; VERIFY THE LAST SECTOR
4191 <1> ;; INT 13H
4192 <1> ;; jnc short RES_OK ; VERIFY OK
4193 <1> ;; cmp AH,BAD_SECTOR ; OK ALSO IF JUST ID READ
4194 <1> ;; JE short RES_OK
4195 <1> ;; cmp AH,DATA_CORRECTED
4196 <1> ;; JE short RES_OK
4197 <1> ;; cmp AH,BAD_ECC
4198 <1> ;; JE short RES_OK
4199 <1> ;; call POD_TCHK ; CHECK FOR TIME OUT
4200 <1> ;; cmp word [wait_count], 6*182 ; waiting time (in timer ticks)
4201 <1> ;; ; (60 seconds)
4202 <1> ;; cmc
4203 <1> ;; jc short RES_ER ; FAILED
4204 <1> ;; mov CX,[NEC_STATUS] ; GET SECTOR ADDRESS, AND CYLINDER
4205 <1> ;; mov AL,CL ; SEPARATE OUT SECTOR NUMBER
4206 <1> ;; and AL,3FH
4207 <1> ;; dec AL ; TRY PREVIOUS ONE
4208 <1> ;; jz short RES_RS ; WE'VE TRIED ALL SECTORS ON TRACK
4209 <1> ;; and CL,0C0H ; KEEP CYLINDER BITS
4210 <1> ;; OR CL,AL ; MERGE SECTOR WITH CYLINDER BITS
4211 <1> ;; mov [NEC_STATUS],CX ; SAVE CYLINDER, NEW SECTOR NUMBER
4212 <1> ;; jmp short RES_3 ; TRY AGAIN
4213 <1> ;; RES_ER: mov SI,OFFSET F1791 ; INDICATE DISK 1 ERROR
4214 <1> ;; ;test DL,1
4215 <1> ;; ;jnz short RES_E1
4216 <1> ;; ;mov SI,OFFSET F1790 ; INDICATE DISK 0 ERROR
4217 <1> ;; RES_E1:
4218 <1> ;; ;call E_MSG ; DISPLAY ERROR AND SET (BP) ERROR FLAG
4219 <1> ;; RES_OK:
4220 <1> ;; ;pop CX ; RESTORE TIMER LIMITS
4221 <1> ;; ;pop BX
4222 <1> ;; ;retn
4223 <1>
4224 <1> ;; SET_FAIL:
4225 <1> ;; ;mov AX,X*(CMOS_DIAG+NMI) ; GET CMOS ERROR BYTE
4226 <1> ;; ;call CMOS_READ
4227 <1> ;; ;OR AL,HF_FAIL ; SET DO NOT IPL FROM DISK FLAG
4228 <1> ;; ;xchg AH,AL ; SAVE IT
4229 <1> ;; ;call CMOS_WRITE ; PUT IT OUT
4230 <1> ;; ;retn
4231 <1>
4232 <1> ;; POD_TCHK: ; CHECK FOR 30 SECOND TIME OUT
4233 <1> ;; ;pop AX ; SAVE RETURN
4234 <1> ;; ;pop CX ; GET TIME OUT LIMITS
4235 <1> ;; ;pop BX
4236 <1> ;; ;push BX ; AND SAVE THEM AGAIN
4237 <1> ;; ;push CX
4238 <1> ;; ;push AX
4239 <1> ;; ;push ds
4240 <1> ;; ;xor ax, ax
4241 <1> ;; ;mov ds, ax ; RESTORE RETURN ; AX = CURRENT TIME
4242 <1> ;; ;mov AX, [TIMER_LOW] ; BX = START TIME
4243 <1> ;; ; ; CX = END TIME
4244 <1> ;; ;
4245 <1> ;; ;pop ds
4246 <1> ;; ;cmp BX,CX
4247 <1> ;; ;JB short TCHK1 ; START < END
4248 <1> ;; ;cmp BX,AX
4249 <1> ;; ;JB short TCHKG ; END < START < CURRENT
4250 <1> ;; ;jmp SHORT TCHK2 ; END, CURRENT < START
4251 <1> ;; TCHK1: cmp AX,BX
4252 <1> ;; ;JB short TCHKNG ; CURRENT < START < END
4253 <1> ;; TCHK2: cmp AX,CX
4254 <1> ;; ;JB short TCHKG ; START < CURRENT < END
4255 <1> ;; ; OR CURRENT < END < START
4256 <1> ;; TCHKNG: STC ; CARRY SET INDICATES TIME OUT
4257 <1> ;; ;retn
4258 <1> ;; TCHKG: CLC ; INDICATE STILL TIME
4259 <1> ;; ;retn
4260 <1>
4261 <1> ;; int_13h:
4262 <1>
4263 <1>
4264 <1> ;; -----
4265 <1> ;; FIXED DISK BIOS ENTRY POINT :
4266 <1> ;; -----
4267 <1> ;; 17/07/2022
4268 <1> ;; 16/07/2022
4269 <1> ;; 13/07/2022 - TRDOS 386 v2.0.5
4270 <1> ;; 15/01/2017
4271 <1> ;; 14/01/2017
4272 <1> ;; 07/01/2017
4273 <1> ;; 02/01/2017
4274 <1> ;; 01/06/2016
4275 <1> ;; 16/05/2016, 27/05/2016, 28/05/2016, 29/05/2016
4276 <1> ;; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
4277 <1> int33h: ; DISK I/O
4278 <1> ; 29/05/2016
4279 00004F42 80642408FE <1> and byte [esp+8], 11111110b ; clear carry bit of eflags register
4280 <1>
4281 <1> ; 13/07/2022
4282 00004F47 06 <1> push es
4283 <1> ; 16/05/2016
4284 00004F48 1E <1> push ds
4285 00004F49 53 <1> push ebx ; user's buffer address (virtual)
4286 <1> ; 13/07/2022
4287 00004F4A 51 <1> push ecx
4288 00004F4B 52 <1> push edx
4289 00004F4C 56 <1> push esi
4290 00004F4D 57 <1> push edi
4291 <1>
4292 <1> ;mov bx, KDATA ; System (kernel's) data segment
4293 <1> ;mov ds, bx
4294 <1> ; 13/07/2022
4295 00004F4E 66BF1000 <1> mov di, KDATA
4296 00004F52 8EDF <1> mov ds, di
4297 00004F54 8EC7 <1> mov es, di
4298 <1>
4299 <1> ;15/01/2017
4300 <1> ; 14/01/2017
4301 <1> ; 02/01/2017
4302 <1> ;mov byte [intflg], 33h ; disk io interrupt
4303 <1> ;pop ebx
4304 <1>
4305 <1> ; 13/07/2022
4306 <1> ;pop dword [user_buffer] ; 01/06/2016
4307 00004F56 891D[2C830100] <1> mov [user_buffer], ebx
4308 <1>
4309 00004F5C C605[667C0100]00 <1> mov byte [scount], 0 ; sector count for transfer
4310 00004F63 80FC03 <1> ah, 03h ; chs write
4311 00004F66 773C <1> ja short int33h_2

```

```

4312 00004F68 7407      <1>      je      short int33h_0
4313 00004F6A 80FC02    <1>      cmp     ah, 02h ; chs read
4314 00004F6D 7267      <1>      jb      short int33h_5
4315 00004F6F EB5A      <1>      jmp     short int33h_4
4316      <1> int33h_0:
4317      <1>      ;; 17/07/2022 - 64K r/w buffer limit check ?
4318      <1>      ;cmp     al, 80h ; 128
4319      <1>      ;ja      short int33h_8 ; error
4320      <1>      ;; 17/07/2022 - zero r/w count check ?
4321      <1>      ;or      al, al
4322      <1>      ;jz      short int33h_8 ; error
4323      <1>
4324      <1>      ; 17/07/2022 (buffer limit and zero count check)
4325 00004F71 FEC8      <1>      dec     al
4326 00004F73 781A      <1>      js      short int33h_8 ; error
4327 00004F75 FEC0      <1>      inc     al
4328      <1>
4329      <1>      ; transfer user's buffer content to sector buffer
4330 00004F77 51        <1>      push    ecx
4331 00004F78 0FB6C8    <1>      movzx   ecx, al
4332      <1> int33h_1:
4333      <1>      ; 13/07/2022
4334      <1>      ;push    esi
4335      <1>      ;mov     esi, [user_buffer]
4336 00004F7B 89DE      <1>      mov     esi, ebx
4337      <1>      ; esi = user's buffer address (virtual, ebx)
4338      <1>      ;push    edi
4339      <1>      ;push    es
4340 00004F7D 50        <1>      push    eax
4341      <1>      ;mov     ax, KDATA
4342      <1>      ;mov     es, ax
4343 00004F7E BF00000700 <1>      mov     edi, Cluster_Buffer
4344 00004F83 C1E109    <1>      shl     ecx, 9 ; * 512
4345 00004F86 E8CABB0000 <1>      call    transfer_from_user_buffer
4346      <1>      ; (ecx and eax will be modified)
4347 00004F8B 58        <1>      pop     eax
4348      <1>      ; 13/07/2022
4349      <1>      ;pop     es
4350      <1>      ;pop     edi
4351      <1>      ;pop     esi
4352 00004F8C 59        <1>      pop     ecx
4353 00004F8D 7347      <1>      jnc     short int33h_5
4354      <1>
4355      <1>      ;mov     ebx, [user_buffer] ; 01/06/2016
4356      <1>      ;pop     ds
4357      <1>
4358      <1> int33h_8:
4359      <1>      ; 13/07/2022
4360 00004F8F B8FF000000 <1>      mov     eax, 0FFh ; Unknown error !?
4361      <1> int33h_9:
4362      <1>      pop     edi
4363      <1>      pop     esi
4364      <1>      pop     edx
4365      <1>      pop     ecx
4366      <1>      pop     ebx
4367      <1>      pop     ds
4368      <1>      pop     es
4369      <1>
4370      <1>      ; 13/07/2022
4371 00004F9B 7305      <1>      jnc     short int33h_7
4372      <1>
4373      <1>      ;;15/01/2017
4374      <1>      ; 02/01/2017
4375      <1>      ;cli
4376      <1>      ;;mov     byte [ss:intflg], 0 ; 07/01/2017
4377      <1>      ;
4378      <1>      ; (*) 29/05/2016
4379      <1>      ; (*) retf 4 ; skip eflags on stack
4380      <1>
4381      <1>      ; 29/05/2016 -set carry flag on stack-
4382      <1>      ; [esp] = EIP
4383      <1>      ; [esp+4] = CS
4384      <1>      ; [esp+8] = E-FLAGS
4385 00004F9D 804C240801 <1>      or      byte [esp+8], 1 ; set carry bit of eflags register
4386      <1>      ; [esp+12] = ESP (user)
4387      <1>      ; [esp+16] = SS (user)
4388      <1>      ;
4389      <1>      ; 13/07/2022
4390      <1> int33h_7:
4391 00004FA2 FA        <1>      cli
4392      <1>      ;;15/01/2017
4393      <1>      ;;mov     byte [ss:intflg], 0 ; 07/01/2017
4394      <1>      ; cf = 0 ; use eflags which is in stack
4395 00004FA3 CF        <1>      iretd
4396      <1>
4397      <1>      ; (*) 29/05/2016 - 'retf 4' instruction causes to stack fault
4398      <1>      ; (OUTER-PRIVILEGE-LEVEL)
4399      <1>      ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986
4400      <1>      ; // RETF instruction:
4401      <1>      ;
4402      <1>      ; IF OperandMode=32 THEN
4403      <1>      ;     Load CS:EIP from stack;
4404      <1>      ;     Set CS RPL to CPL;
4405      <1>      ;     Increment ESP by 8 plus the immediate offset if it exists;
4406      <1>      ;     Load SS:ESP from stack;
4407      <1>      ; ELSE (* OperandMode=16 *)
4408      <1>      ;     Load CS:IP from stack;
4409      <1>      ;     Set CS RPL to CPL;
4410      <1>      ;     Increment ESP by 4 plus the immediate offset if it exists;
4411      <1>      ;     Load SS:ESP from stack;
4412      <1>      ; FI;
4413      <1>      ;
4414      <1>      ; //
4415      <1>
4416      <1> int33h_2:
4417 00004FA4 80FC05    <1>      cmp     ah, 05h ; format track
4418 00004FA7 7709      <1>      ja      short int33h_3
4419 00004FA9 722B      <1>      jb      short int33h_5
4420 00004FAB 51        <1>      push    ecx
4421      <1>      ;mov     ecx, 1
4422      <1>      ; 17/07/2022
4423 00004FAC 31C9      <1>      xor     ecx, ecx
4424 00004FAE FEC1      <1>      inc     cl
4425      <1>      ; ecx = 1
4426 00004FB0 EBC9      <1>      jmp     short int33h_1
4427      <1> int33h_3:
4428 00004FB2 80FC1C    <1>      cmp     ah, 1Ch ; LBA write
4429 00004FB5 771F      <1>      ja      short int33h_5
4430 00004FB7 74B8      <1>      je      short int33h_0
4431 00004FB9 80FC1B    <1>      cmp     ah, 1Bh ; LBA read
4432 00004FBC 740D      <1>      je      short int33h_4
4433 00004FBE 80FC08    <1>      cmp     ah, 08h ; get disk parameters
4434 00004FC1 7513      <1>      jne     short int33h_5
4435      <1>      ; 01/06/2016

```

```

4436 00004FC3 8B1D[2C830100] <1> mov ebx, [user_buffer] ; user's buffer address
4437 00004FC9 EB10 <1> jmp short int33h_6
4438 <1> int33h_4:
4439 <1> ;; 17/07/2022 - 64K r/w buffer limit check ?
4440 <1> ;cmp al, 80h ; 128
4441 <1> ;ja short int33h_8 ; error
4442 <1> ;; 17/07/2022 - zero r/w count check ?
4443 <1> ;or al, al
4444 <1> ;jz short int33h_8 ; error
4445 <1>
4446 <1> ; 17/07/2022 (buffer limit and zero count check)
4447 00004FCB FEC8 <1> dec al
4448 00004FCD 78C0 <1> js short int33h_8 ; error
4449 00004FCF FEC0 <1> inc al
4450 <1>
4451 00004FD1 A2[667C0100] <1> mov byte [scount], al ; <= 128 sectors
4452 <1> int33h_5:
4453 00004FD6 BB00000700 <1> mov ebx, Cluster_Buffer ; max. 65536 bytes
4454 <1> ; buf. addr: 70000h
4455 <1> ;mov byte [ClusterBuffer_Valid], 0
4456 <1> int33h_6:
4457 <1> ; 13/07/2022
4458 <1> ;pop ds
4459 <1> ;pushfd
4460 <1> ;push cs
4461 <1>
4462 00004FDB E83B000000 <1> call DISK_IO
4463 <1>
4464 <1> ;;mov ebx, [cs:user_buffer] ; 01/06/2016
4465 <1> ;mov ebx, [user_buffer] ; 13/07/2022
4466 00004FE0 72B2 <1> jc short int33h_9
4467 <1>
4468 <1> ;cmp byte [cs:scount], 0
4469 <1> ;jna short int33h_7
4470 00004FE2 803D[667C0100]00 <1> cmp byte [scount], 0 ; 13/07/2022
4471 00004FE9 76A9 <1> jna short int33h_9
4472 <1>
4473 <1> ; 13/07/2022
4474 <1>
4475 <1> ; transfer sector buffer content to user's buffer
4476 <1> ;push es
4477 <1> ;push ds
4478 <1> ;push eax
4479 <1> ;mov ax, KDATA
4480 <1> ;mov ds, ax
4481 <1> ;mov es, ax
4482 <1> ;push ecx
4483 <1> ;push esi
4484 <1> ;push edi
4485 <1>
4486 <1> ; 13/07/2022
4487 00004FEB 50 <1> push eax
4488 00004FEC 0FB60D[667C0100] <1> movzx ecx, byte [scount]
4489 00004FF3 C1E109 <1> shl ecx, 9 ; * 512 bytes
4490 <1> ;mov edi, ebx ; user's buffer address
4491 00004FF6 8B3D[2C830100] <1> mov edi, [user_buffer] ; 13/07/2022
4492 00004FFC BE00000700 <1> mov esi, Cluster_Buffer
4493 00005001 E805B80000 <1> call transfer_to_user_buffer
4494 <1> ; (ecx and eax will be modified)
4495 00005006 58 <1> pop eax
4496 <1>
4497 <1> ; 13/07/2022
4498 00005007 7286 <1> jc short int33h_8 ; eax = 0FFh
4499 00005009 EB89 <1> jmp short int33h_9 ; cf = 0
4500 <1>
4501 <1> ;pop edi
4502 <1> ;pop esi
4503 <1> ;pop ecx
4504 <1> ;pop eax
4505 <1> ;pop ds
4506 <1> ;pop es
4507 <1> ;jc short int33h_8
4508 <1> ;int33h_7:
4509 <1> ;cli
4510 <1> ;;15/01/2017
4511 <1> ;;mov byte [ss:intflg], 0 ; 07/01/2017
4512 <1> ; cf = 0 ; use eflags which is in stack
4513 <1> ;iretd
4514 <1> ;int33h_8:
4515 <1> ;mov eax, 0FFh ; Unknown error !?
4516 <1> ; 13/07/2022
4517 <1> ;xor eax, eax
4518 <1> ;dec al ; eax = 0FFh
4519 <1> ;jmp short int33h_9
4520 <1>
4521 <1> ;int33h_9:
4522 <1> ;; cf = 1
4523 <1> ;
4524 <1> ;; (*) 29/05/2016
4525 <1> ;; (*) retf 4 ; skip eflags on stack
4526 <1> ;; Note: This 'retf 4' was wrong, -it was causing
4527 <1> ;; to stack errors in ring 3-
4528 <1> ;; POP sequence of 'retf 4' is as
4529 <1> ;; "eip, cs, eflags, esp, ss, +4 bytes"
4530 <1> ;; it is not as "eip, cs, +4 bytes, esp, ss" !
4531 <1> ;
4532 <1> ;; 29/05/2016 -set carry flag on stack-
4533 <1> ;or byte [esp+8], 1 ; set carry bit of eflags register
4534 <1> ;iretd
4535 <1> ;jmp short int33h_7 ; 07/01/2017
4536 <1>
4537 <1> ;; 11/04/2021
4538 <1> ;int13h: ; 21/02/2015
4539 <1> ; cll ; 11/04/2021
4540 <1> ; pushfd
4541 <1> ; push cs
4542 <1> ; call DISK_IO
4543 <1> ; ret
4544 <1>
4545 <1> int13h:
4546 <1> ; 13/07/2022 - TRDOS 386 v2.0.5
4547 <1> ; Note: DISK_IO sets registers on stack
4548 <1> ; as return parameters. So,
4549 <1> ; stack order (at the entry of 'DISK_IO')
4550 <1> ; must be same with 'int33h:' as above.
4551 <1>
4552 <1> ;push es ; not necessary
4553 <1> ;push ds ; not necessary
4554 <1> ;
4555 <1> ; following pushes are necessary
4556 <1> ; for setting registers -return values- in DISK_IO
4557 0000500B 53 <1> push ebx
4558 0000500C 51 <1> push ecx
4559 0000500D 52 <1> push edx

```

```

4560 0000500E 56      <1>      push    esi
4561 0000500F 57      <1>      push    edi
4562                <1>      ;push    ebp
4563                <1>      ;mov     ebp, esp
4564                <1>      ; edi = ebp+4
4565                <1>      ; esi = ebp+8
4566                <1>      ; edx = ebp+12
4567                <1>      ; ecx = ebp+16
4568                <1>      ; ebx = ebp+20
4569                <1>      ;
4570 00005010 E806000000 <1>      call    DISK_IO
4571                <1>      ;
4572 00005015 5F      <1>      pop     edi
4573 00005016 5E      <1>      pop     esi
4574 00005017 5A      <1>      pop     edx
4575 00005018 59      <1>      pop     ecx
4576 00005019 5B      <1>      pop     ebx
4577                <1>      ;
4578                <1>      ;pop     ds
4579                <1>      ;pop     es
4580                <1>      ;
4581 0000501A C3      <1>      retn
4582                <1>      ;
4583                <1>      ; 10/08/2022
4584                <1>      ; 07/08/2022
4585                <1>      ; 17/07/2022
4586                <1>      ; 13/07/2022 - TRDOS 386 v2.0.5
4587                <1>      ; 18/04/2021 - TRDOS 386 v2.0.4
4588                <1>      ; 11/04/2021 - TRDOS 386 v2.0.3
4589                <1>      ; 30/08/2020
4590                <1>      ; 09/12/2017
4591                <1>      ; 29/05/2016
4592                <1>      ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
4593                <1>      ;
4594                <1>      DISK_IO:
4595                <1>      ; 10/08/2022
4596                <1>      ; 17/07/2022
4597                <1>      ; 13/07/2022
4598                <1>      ; Registers are also on stack
4599                <1>      ; (with same contents)
4600                <1>      ; in following order:
4601                <1>      ;
4602                <1>      ;     ebx = esp+20
4603                <1>      ;     ecx = esp+16
4604                <1>      ;     edx = esp+12
4605                <1>      ;     esi = esp+8
4606                <1>      ;     edi = esp+4
4607                <1>      ;
4608                <1>      ; cs = KCODE (== KDATA base address)
4609                <1>      ; ss = KDATA
4610                <1>      ; ds = KDATA
4611                <1>      ; es = KDATA
4612                <1>      ;
4613                <1>      ; 17/07/2022
4614 0000501B FB      <1>      sti             ; ENABLE INTERRUPTS
4615                <1>      ;
4616 0000501C 80FA80 <1>      cmp     dl, 80h          ; TEST FOR FIXED DISK DRIVE
4617                <1>      ;jae     short A1          ; YES, HANDLE HERE
4618                <1>      ;;;int 40h          ; DISKETTE HANDLER
4619                <1>      ;;;call int40h
4620                <1>      ;jb     DISKETTE_IO_1
4621                <1>      ;RET_2:
4622                <1>      ;;;retf 2          ; BACK TO CALLER
4623                <1>      ;;;retf 4
4624                <1>      ; 11/04/2021
4625 0000501F 7305 <1>      jnb     short A1
4626 00005021 E934F2FFFF <1>      jmp     DISKETTE_IO_1
4627                <1>      A1:
4628                <1>      ;sti             ; 17/07/2022          ; ENABLE INTERRUPTS
4629                <1>      ;
4630                <1>      ;;; 04/01/2015
4631                <1>      ;;;or     ah, ah
4632                <1>      ;;;jnz     short A2
4633                <1>      ;;;int 40h          ; RESET NEC WHEN AH=0
4634                <1>      ;;;sub     ah, ah
4635                <1>      ;
4636 00005026 80FA83 <1>      cmp     dl, (80h + S_MAX_FILE - 1)
4637                <1>      ;ja     short RET_2
4638 00005029 760A <1>      jna     short _A0
4639                <1>      ;
4640                <1>      ; 13/07/2022
4641                <1>      ; (here, DS is KDATA segment already)
4642                <1>      ;
4643                <1>      ; 29/05/2016
4644                <1>      ;push    ds
4645                <1>      ; 11/04/2021
4646                <1>      ;push    eax
4647                <1>      ;mov     ax, KDATA
4648                <1>      ;mov     ds, ax
4649                <1>      ; 11/04/2021
4650                <1>      ;pop     eax
4651                <1>      ;
4652 0000502B B4AA <1>      mov     ah, 0AAh          ; Hard disk drive not ready !
4653                <1>      ; (Programmer's guide to AMIBIOS, 1992)
4654 0000502D 8825[37770100] <1>      mov     byte [DISK_STATUS1], ah
4655                <1>      ; 13/07/2022
4656                <1>      ;pop     ds
4657                <1>      ;jmp     short RET_2
4658 00005033 F9 <1>      stc
4659 00005034 C3 <1>      retn
4660                <1>      _A0:
4661                <1>      ; 18/01/2015
4662 00005035 08E4 <1>      or     ah, ah
4663 00005037 742C <1>      jz     short A4
4664 00005039 80FC0D <1>      cmp     ah, 0Dh ; Alternate reset
4665 0000503C 7504 <1>      jne     short A2
4666 0000503E 28E4 <1>      sub     ah, ah ; Reset
4667 00005040 EB23 <1>      jmp     short A4
4668                <1>      A2:
4669                <1>      ; 13/07/2022
4670 00005042 80FC08 <1>      cmp     ah, 08h          ; GET PARAMETERS IS A SPECIAL CASE
4671 00005045 7505 <1>      jne     short A3
4672 00005047 E911040000 <1>      jmp     GET_PARM_N
4673                <1>      A3:
4674                <1>      ; 13/07/2022
4675 0000504C 80FC15 <1>      cmp     ah, 15h          ; READ DASD TYPE IS ALSO
4676 0000504F 7514 <1>      jne     short A4
4677 00005051 E9AC030000 <1>      jmp     READ_DASD_TYPE ; Return Drive Type
4678                <1>      ; (Programmer's guide to AMIBIOS, 1992)
4679                <1>      ; 13/07/2022
4680                <1>      int33h_bad_cmd:
4681                <1>      ; 16/05/2016
4682                <1>      ; 30/01/2015
4683                <1>      ; 29/05/2016

```



```

4684      <1>      ;push  ds
4685      <1>      ;push  eax
4686      <1>      ;mov   ax, KDATA
4687      <1>      ;mov   ds, ax
4688      <1>      ;pop   eax
4689
4690 00005056 B401      <1>      mov   ah, BAD_CMD
4691 00005058 8825[37770100] <1>      mov   [DISK_STATUS1], ah ; BAD_CMD ; COMMAND ERROR
4692      <1>      ;jmp    short RET_2
4693      <1>      ; 13/07/2022
4694      <1>      ;RET_2:
4695      <1>      ; (*) 29/05/2016
4696      <1>      ; (*) retf 4
4697      <1>      ;or    byte [esp+8], 1 ; set carry bit of eflags register
4698      <1>      ;iretd
4699      <1>
4700      <1>      ; 13/07/2022
4701 0000505E F9      <1>      stc
4702      <1>      ; cf = 1, ah = BAD_CMD
4703 0000505F C3      <1>      retn
4704      <1>      _A4:
4705      <1>      ; 13/07/2022
4706      <1>      ; 02/02/2015
4707 00005060 80FC1D    <1>      cmp    ah, 1Dh          ; (Temporary for Retro UNIX 386 v1)
4708      <1>      ; 12/01/2015
4709      <1>      ;cmc
4710      <1>      ;jnc    short A4
4711      <1>      ; 13/07/2022
4712 00005063 73F1      <1>      jnb    short int33h_bad_cmd
4713      <1>      A4:
4714 00005065 C8080000 <1>      enter   8, 0          ; SAVE REGISTERS DURING OPERATION
4715      <1>                                     ; SAVE (BP) AND MAKE ROOM FOR @CMD_BLOCK
4716      <1>      ; 13/07/2022
4717      <1>      ; ENTER 8, 0
4718      <1>      ;push  ebp
4719      <1>      ;mov   ebp, esp
4720      <1>      ;sub   esp, 8
4721      <1>      ;
4722      <1>      ;push  ebx          ; IN THE STACK, THE COMMAND BLOCK IS:
4723      <1>      ;push  ecx          ; @CMD_BLOCK == BYTE PTR [BP]-8
4724      <1>      ;push  edx
4725      <1>      ;push  esi
4726      <1>      ;push  edi
4727      <1>
4728      <1>      ; 13/07/2022
4729      <1>      ; edi = ebp+8
4730      <1>      ; esi = ebp+12
4731      <1>      ; edx = ebp+16
4732      <1>      ; ecx = ebp+20
4733      <1>      ; ebx = ebp+24
4734      <1>
4735      <1>      ;;04/01/2015
4736      <1>      ;or    ah, ah          ; CHECK FOR RESET
4737      <1>      ;jnz   short A5
4738      <1>      ;mov   dl, 80h          ; FORCE DRIVE 80 FOR RESET
4739      <1>      ;;A5:
4740      <1>      ; 13/07/2022
4741 00005069 E880000000 <1>      call   DISK_IO_CONT      ; PERFORM THE OPERATION
4742      <1>      ;;call  DDS          ; ESTABLISH SEGMENT
4743 0000506E 8A25[37770100] <1>      mov    ah, [DISK_STATUS1] ; GET STATUS FROM OPERATION
4744      <1>      ;(*) cmp ah, 1          ; SET THE CARRY FLAG TO INDICATE
4745      <1>                                     ; SUCCESS OR FAILURE
4746      <1>      ;pop   edi          ; RESTORE REGISTERS
4747      <1>      ;pop   esi
4748      <1>      ;pop   edx
4749      <1>      ;pop   ecx
4750      <1>      ;pop   ebx
4751      <1>
4752 00005074 C9      <1>      leave          ; ADJUST (SP) AND RESTORE (BP)
4753      <1>
4754      <1>      ;retf 2          ; THROW AWAY SAVED FLAGS
4755      <1>      ; (*) 29/05/2016
4756      <1>      ; (*) retf 4
4757      <1>
4758      <1>      ; 13/07/2022
4759 00005075 80FC01    <1>      cmp    ah, 1
4760      <1>      ;jc    short _A5
4761      <1>      ;or    byte [esp+8], 1 ; set carry bit of eflags register
4762      <1>      ;_A5:
4763      <1>      ;iretd
4764      <1>      ; 10/08/2022
4765 00005078 F5      <1>      cmc
4766      <1>      ; 13/07/2022
4767 00005079 C3      <1>      retn
4768      <1>
4769      <1>      ; 21/02/2015
4770      <1>      ; dw --> dd
4771      <1>      ; 13/07/2022
4772      <1>      D1:
4773 0000507A [91520000] <1>      dd     DISK_RESET          ; FUNCTION TRANSFER TABLE
4774 0000507E [F1520000] <1>      dd     RETURN_STATUS        ; 00h
4775 00005082 [04530000] <1>      dd     DISK_READ            ; 01h
4776 00005086 [60530000] <1>      dd     DISK_WRITE          ; 02h
4777 0000508A [EA530000] <1>      dd     DISK_VERF           ; 03h
4778 0000508E [D4530000] <1>      dd     FMT_TRK            ; 04h
4779 00005092 [27520000] <1>      dd     BAD_COMMAND         ; 05h
4780 00005096 [27520000] <1>      dd     BAD_COMMAND         ; 06h FORMAT BAD SECTORS
4781 0000509A [27520000] <1>      dd     BAD_COMMAND         ; 07h FORMAT DRIVE
4782 0000509E [DD540000] <1>      dd     INIT_DRV            ; 08h RETURN PARAMETERS
4783 000050A2 [FE520000] <1>      dd     RD_LONG             ; 09h
4784 000050A6 [5A530000] <1>      dd     WR_LONG             ; 0Ah
4785 000050AA [4E550000] <1>      dd     DISK_SEEK          ; 0Bh
4786 000050AE [91520000] <1>      dd     DISK_RESET          ; 0Ch
4787 000050B2 [27520000] <1>      dd     BAD_COMMAND         ; 0Dh
4788 000050B6 [27520000] <1>      dd     BAD_COMMAND         ; 0Eh READ BUFFER
4789 000050BA [76550000] <1>      dd     TST_RDY            ; 0Fh WRITE BUFFER
4790 000050BE [AA550000] <1>      dd     HDISK_RECAL          ; 10h
4791 000050C2 [27520000] <1>      dd     BAD_COMMAND         ; 11h
4792 000050C6 [27520000] <1>      dd     BAD_COMMAND         ; 12h MEMORY DIAGNOSTIC
4793 000050CA [E0550000] <1>      dd     CTLR_DIAGNOSTIC        ; 13h DRIVE DIAGNOSTIC
4794      <1>      ;; 02/02/2015 (Temporary - Retro UNIX 386 v1 - DISK I/O test)
4795 000050CE [27520000] <1>      dd     BAD_COMMAND         ; 14h CONTROLLER DIAGNOSTIC
4796 000050D2 [27520000] <1>      dd     BAD_COMMAND         ; 15h
4797 000050D6 [27520000] <1>      dd     BAD_COMMAND         ; 16h
4798 000050DA [27520000] <1>      dd     BAD_COMMAND         ; 17h
4799 000050DE [27520000] <1>      dd     BAD_COMMAND         ; 18h
4800 000050E2 [27520000] <1>      dd     BAD_COMMAND         ; 19h
4801 000050E6 [04530000] <1>      dd     DISK_READ            ; 1Ah
4802 000050EA [60530000] <1>      dd     DISK_WRITE          ; 1Bh ; LBA read
4803      <1>      D1L EQU $ - D1          ; 1Ch ; LBA write
4804      <1>
4805      <1>      ; 02/12/2023
4806      <1>      ; 01/12/2023 - TRDOS 386 v2.0.7
4807      <1>      ; 07/08/2022

```

```

4808      <1>      ; 17/07/2022 - TRDOS 386 v2.0.5
4809      <1> DISK_IO_CONT:
4810      <1>      ;;call DDS      ; ESTABLISH SEGMENT
4811      <1>      ; 11/04/2021
4812      <1>      cmp      ah, 01h      ; RETURN STATUS
4813      <1>      jne      short SU0
4814      <1>      jmp      RETURN_STATUS
4815      <1> SU0:
4816      <1>      mov      byte [DISK_STATUS1], 0      ; RESET THE STATUS INDICATOR
4817      <1>      ; 13/07/2022
4818      <1>      mov      esi, ebx ; 21/02/2015 ; SAVE DATA ADDRESS
4819      <1>      mov      bl, [HF_NUM]      ; GET NUMBER OF DRIVES
4820      <1>      and      dl, 7Fh      ; GET DRIVE AS 0 OR 1
4821      <1>      ; (get drive number as 0 to 3)
4822      <1>      ; 14/02/2015
4823      <1>      cmp      bl, dl
4824      <1>      jbe      short BAD_COMMAND      ; INVALID DRIVE
4825      <1>      ; 07/08/2022
4826      <1>      ja      short SU0X
4827      <1>      jmp      BAD_COMMAND
4828      <1> SU0X:
4829      <1>      ;;03/01/2015
4830      <1>      sub      ebx, ebx
4831      <1>      mov      bl, dl
4832      <1>      mov      [LBAMode], bh ; 0
4833      <1>
4834      <1>      ;test byte [ebx+hd0_type], 1      ; LBA ready ?
4835      <1>      ;jz short su1      ; no
4836      <1>      ;inc byte [LBAMode]
4837      <1> ;su1:
4838      <1>      ; 11/04/2021 (32 bit push/pop)
4839      <1>      ; 21/02/2015 (32 bit modification)
4840      <1>      ; 04/01/2015
4841      <1>      push     eax ; ***
4842      <1>      ;push es ; **
4843      <1>      push     edx ; *
4844      <1>      push     eax ; ****
4845      <1>      call     GET_VEC      ; GET DISK PARAMETERS
4846      <1>      ; 02/02/2015
4847      <1>      ;mov ax, [ES:BX+16] ; I/O port base address (1F0h, 170h)
4848      <1>      mov      ax, [ebx+16]
4849      <1>      mov      [HF_PORT], ax
4850      <1>      ;mov dx, [ES:BX+18] ; control port address (3F6h, 376h)
4851      <1>      mov      dx, [ebx+18]
4852      <1>      mov      [HF_REG_PORT], dx
4853      <1>      ;mov al, [ES:BX+20] ; head register upper nibble (A0h,B0h,E0h,F0h)
4854      <1>      mov      al, [ebx+20]
4855      <1>      ; 23/02/2015
4856      <1>      test     al, 40h ; LBA bit (bit 6)
4857      <1>      jz      short su1
4858      <1>      inc      byte [LBAMode] ; 1
4859      <1> su1:
4860      <1>      shr      al, 4
4861      <1>      and      al, 1
4862      <1>      mov      [hf_m_s], al
4863      <1>      ;
4864      <1>      ; 03/01/2015
4865      <1>      ;mov al, [ES:BX+8]      ; GET CONTROL BYTE MODIFIER
4866      <1>      mov      al, [ebx+8]
4867      <1>      ;mov dx, [HF_REG_PORT]      ; Device Control register
4868      <1>      out      dx, al      ; SET EXTRA HEAD OPTION
4869      <1>      ;-here-
4870      <1>      ; Control Byte: (= 08h)
4871      <1>      ; bit 0 - 0
4872      <1>      ; bit 1 - NIEN (1 = disable irq)
4873      <1>      ; bit 2 - SRST (software RESET)
4874      <1>      ; bit 3 - use extra heads (8 to 15)
4875      <1>      ; -always set to 1-
4876      <1>      ; (bits 3 to 7 are reserved
4877      <1>      ; for ATA devices)
4878      <1>      mov      ah, [CONTROL_BYTE]      ; SET EXTRA HEAD OPTION IN
4879      <1>      and      ah, 0C0h      ; CONTROL BYTE
4880      <1>      or      ah, al
4881      <1>      mov      [CONTROL_BYTE], ah
4882      <1>
4883      <1>      ; 11/04/2021 (32 bit push/pop)
4884      <1>      ; 04/01/2015
4885      <1>      pop      eax ; ****
4886      <1>      pop      edx ; * ; 14/02/2015
4887      <1>      and      ah, ah ; Reset function ?
4888      <1>      jnz      short su2
4889      <1>      ;pop es ; **
4890      <1>      pop      eax ; ***
4891      <1>      jmp      DISK_RESET
4892      <1> su2:
4893      <1>      cmp      byte [LBAMode], 0
4894      <1>      jna      short su3
4895      <1>      ;
4896      <1>      ; 02/02/2015 (LBA read/write function calls)
4897      <1>      cmp      ah, 1Bh
4898      <1>      jb      short lbarw1
4899      <1>      cmp      ah, 1Ch
4900      <1>      ja      short invldfnc
4901      <1>      ;pop edx ; * ; 14/02/2015
4902      <1>      ;mov ax, cx ; Lower word of LBA address (bits 0-15)
4903      <1>
4904      <1>      ; 01/12/2023 (48 bit LBA rw)
4905      <1>      jmp      lba_read_write
4906      <1>
4907      <1> lbarw1:
4908      <1>      ; convert CHS to LBA
4909      <1>      ;
4910      <1>      ; LBA calculation - AWARD BIOS - 1999 - AHDSK.ASM
4911      <1>      ; LBA = "# of Heads" * Sectors/Track * Cylinder + Head * Sectors/Track
4912      <1>      ; + Sector - 1
4913      <1>      ; 11/04/2021 (32 bit push/pop)
4914      <1>      push     edx ; * ; 14/02/2015
4915      <1>      ;xor dh, dh
4916      <1>      xor      edx, edx
4917      <1>      ;mov dl, [ES:BX+14] ; sectors per track (logical)
4918      <1>      mov      dl, [ebx+14]
4919      <1>      ;xor ah, ah
4920      <1>      xor      eax, eax
4921      <1>      ;mov al, [ES:BX+2] ; heads (logical)
4922      <1>      mov      al, [ebx+2]
4923      <1>      dec      al
4924      <1>      inc      ax      ; 0 = 256
4925      <1>      mul      dx
4926      <1>      ; AX = # of Heads * Sectors/Track
4927      <1>      mov      dx, cx
4928      <1>      ;and cx, 3Fh ; sector (1 to 63)
4929      <1>      and      ecx, 3Fh
4930      <1>      xchg     dl, dh
4931      <1>      shr      dh, 6

```

```

4932             <1>             ; DX = cylinder (0 to 1023)
4933             <1>             ;mul     dx
4934             <1>             ; DX:AX = # of Heads" * Sectors/Track * Cylinder
4935 000051A7 F7E2             <1>             mul     edx
4936 000051A9 FEC9             <1>             dec     cl ; sector - 1
4937             <1>             ;add     ax, cx
4938             <1>             ;adc     dx, 0
4939             <1>             ; DX:AX = # of Heads" * Sectors/Track * Cylinder + Sector - 1
4940 000051AB 01C8             <1>             add     eax, ecx
4941             <1>             ; 11/04/2021 (32 bit push/pop)
4942 000051AD 59               <1>             pop     ecx ; * ; ch = head, cl = drive number (zero based)
4943             <1>             ;push    dx
4944             <1>             ;push    ax
4945 000051AE 50               <1>             push   eax
4946             <1>             ; 13/07/2022
4947 000051AF 29C0             <1>             sub     eax, eax
4948             <1>             ;mov     al, [ES:BX+14] ; sectors per track (logical)
4949 000051B1 8A430E           <1>             mov     al, [ebx+14]
4950 000051B4 F6E5             <1>             mul     ch
4951             <1>             ; AX = Head * Sectors/Track
4952             <1>             ; 13/07/2022
4953             <1>             ;movzx   eax, ax ; 09/12/2017
4954             <1>             ;pop     dx
4955 000051B6 5A               <1>             pop     edx
4956             <1>             ;add     ax, dx
4957             <1>             ;pop     dx
4958             <1>             ;adc     dx, 0 ; add carry bit
4959 000051B7 01D0             <1>             add     eax, edx
4960             <1>             lbarw2:
4961 000051B9 29D2             <1>             sub     edx, edx ; 21/02/2015
4962 000051BB 88CA             <1>             mov     dl, cl ; 21/02/2015
4963 000051BD C645F800         <1>             mov     byte [CMD_BLOCK], 0 ; Features Register
4964             <1>             ; NOTE: Features register (1F1h, 171h)
4965             <1>             ; is not used for ATA device R/W functions.
4966             <1>             ; It is old/obsolete 'write precompensation'
4967             <1>             ; register and error register
4968             <1>             ; for old ATA/IDE devices.
4969             <1>             ; 18/01/2014
4970             <1>             ;mov     ch, [hf_m_s] ; Drive 0 (master) or 1 (slave)
4971 000051C1 8A0D[72660000]   <1>             mov     cl, [hf_m_s]
4972             <1>             ;shl     ch, 4 ; bit 4 (drive bit)
4973             <1>             ;or      ch, 0E0h ; bit 5 = 1
4974             <1>             ; ; bit 6 = 1 = LBA mode
4975             <1>             ; ; bit 7 = 1
4976 000051C7 80C90E           <1>             or      cl, 0Eh ; 1110b
4977             <1>             ;and     dh, 0Fh ; LBA byte 4 (bits 24 to 27)
4978 000051CA 25FFFFFF0F       <1>             and     eax, 0FFFFFFFh
4979 000051CF C1E11C           <1>             shl     ecx, 28 ; 21/02/2015
4980             <1>             ;or      dh, ch
4981 000051D2 09C8             <1>             or      eax, ecx
4982             <1>             ;mov     [CMD_BLOCK+2], al ; LBA byte 1 (bits 0 to 7)
4983             <1>             ; ; (Sector Number Register)
4984             <1>             ;mov     [CMD_BLOCK+3], ah ; LBA byte 2 (bits 8 to 15)
4985             <1>             ; ; (Cylinder Low Register)
4986             <1>             ;mov     [CMD_BLOCK+2], ax ; LBA byte 1, 2
4987             <1>             ;mov     [CMD_BLOCK+4], dl ; LBA byte 3 (bits 16 to 23)
4988             <1>             ; ; (Cylinder High Register)
4989             <1>             ;mov     [CMD_BLOCK+5], dh ; LBA byte 4 (bits 24 to 27)
4990             <1>             ; ; (Drive/Head Register)
4991             <1>
4992             <1>             ;mov     [CMD_BLOCK+4], dx ; LBA byte 4, LBA & DEV select bits
4993 000051D4 8945FA           <1>             mov     [CMD_BLOCK+2], eax ; 21/02/2015
4994             <1>             ; 14/02/2015
4995             <1>             ;mov     dl, cl ; Drive number (INIT_DRV)
4996 000051D7 EB36             <1>             jmp     short su4
4997             <1>             su3:
4998             <1>             ; 07/08/2022
4999             <1>             ; 13/07/2022
5000             <1>             ; 02/02/2015
5001             <1>             ; (1Bh & 1Ch functions are not valid for CHS mode)
5002 000051D9 80FC14           <1>             cmp     ah, 14h
5003 000051DC 7603             <1>             jna     short chsfnc
5004             <1>             invldfnc:
5005             <1>             ; 14/02/2015
5006             <1>             ;pop     es ; **
5007             <1>             ; 11/04/2021
5008 000051DE 58               <1>             pop     eax ; ***
5009 000051DF EB46             <1>             jmp     short BAD_COMMAND
5010             <1>             chsfnc:
5011             <1>             ;mov     ax, [ES:BX+5] ; GET WRITE PRE-COMPENSATION CYLINDER
5012 000051E1 668B4305         <1>             mov     ax, [ebx+5]
5013             <1>             ;shr     ax, 2
5014             <1>             ; 07/08/2022
5015 000051E5 C1E802           <1>             shr     eax, 2
5016 000051E8 8845F8           <1>             mov     [CMD_BLOCK], al
5017             <1>
5018             <1>             ;mov     al, [ES:BX+8] ; GET CONTROL BYTE MODIFIER
5019             <1>             ;push    edx ; *
5020             <1>             ;mov     dx, [HF_REG_PORT]
5021             <1>             ;out     dx, al ; SET EXTRA HEAD OPTION
5022             <1>             ;pop     edx ; *
5023             <1>             ;pop     es ; **
5024             <1>             ;mov     ah, [CONTROL_BYTE] ; SET EXTRA HEAD OPTION IN
5025             <1>             ;and     ah, 0C0h ; CONTROL BYTE
5026             <1>             ;or      ah, al
5027             <1>             ;mov     [CONTROL_BYTE], ah
5028             <1>
5029 000051EB 88C8             <1>             mov     al, cl ; GET SECTOR NUMBER
5030 000051ED 243F             <1>             and     al, 3Fh
5031 000051EF 8845FA           <1>             mov     [CMD_BLOCK+2], al
5032 000051F2 886DFB           <1>             mov     [CMD_BLOCK+3], ch ; GET CYLINDER NUMBER
5033 000051F5 88C8             <1>             mov     al, cl
5034 000051F7 C0E806           <1>             shr     al, 6
5035 000051FA 8845FC           <1>             mov     [CMD_BLOCK+4], al ; CYLINDER HIGH ORDER 2 BITS
5036             <1>
5037             <1>             ;05/01/2015
5038             <1>             ;mov     al, dl ; DRIVE NUMBER
5039 000051FD A0[72660000]     <1>             mov     al, [hf_m_s]
5040 00005202 C0E004           <1>             shl     al, 4
5041 00005205 80E60F           <1>             and     dh, 0Fh ; HEAD NUMBER
5042 00005208 08F0             <1>             or      al, dh
5043             <1>             ;or      al, 80h or 20h
5044 0000520A 0CA0             <1>             or      al, 80h+20h ; ECC AND 512 BYTE SECTORS
5045 0000520C 8845FD           <1>             mov     [CMD_BLOCK+5], al ; ECC/SIZE/DRIVE/HEAD
5046             <1>             su4:
5047             <1>             ;pop     es ; **
5048             <1>             ; 14/02/2015
5049             <1>             ;pop     ax
5050             <1>             ;mov     [CMD_BLOCK+1], al ; SECTOR COUNT
5051             <1>             ;push    ax
5052             <1>             ;mov     al, ah ; GET INTO LOW BYTE
5053             <1>             ;xor     ah, ah ; ZERO HIGH BYTE
5054             <1>             ;sal     ax, 1 ; *2 FOR TABLE LOOKUP
5055             <1>             ; 11/04/2021

```

```

5056 0000520F 58          <1>      pop     eax ; ***
5057 00005210 8845F9      <1>      mov     [CMD_BLOCK+1], al
5058 00005213 29DB          <1>      sub     ebx, ebx
5059                      <1>      ;xor     bh, bh
5060 00005215 88E3          <1>      mov     bl, ah
5061                      <1>      ;sal     bx, 1
5062                      <1>      ; 17/07/2022
5063 00005217 C1E302      <1>      sal     ebx, 2 ; 32 bit offset (21/02/2015)
5064                      <1>      ;mov     si, ax ; PUT INTO SI FOR BRANCH
5065                      <1>      ; 13/07/2022
5066                      <1>      ;cmp     bx, D1L ; TEST WITHIN RANGE
5067                      <1>      ;jnb     short BAD_COMMAND_POP
5068 0000521A 83FB74      <1>      cmp     ebx, D1L ; TEST WITHIN RANGE
5069 0000521D 7308          <1>      jnb     short BAD_COMMAND
5070                      <1>      ;xchg     bx, si
5071 0000521F 87F3          <1>      xchg     ebx, esi
5072                      <1>      ;;pop     ax ; RESTORE AX
5073                      <1>      ;;pop     bx ; AND DATA ADDRESS
5074                      <1>
5075                      <1>      ;;push    cx
5076                      <1>      ;;push    ax ; ADJUST ES:BX
5077                      <1>      ;mov     cx, bx ; GET 3 HIGH ORDER NIBBLES OF BX
5078                      <1>      ;shr     cx, 4
5079                      <1>      ;mov     ax, es
5080                      <1>      ;add     ax, cx
5081                      <1>      ;mov     es, ax
5082                      <1>      ;and     bx, 000Fh ; ES:BX CHANGED TO ES:000X
5083                      <1>      ;;pop     ax
5084                      <1>      ;;pop     cx
5085                      <1>      ;;jmp     word [CS:SI+D1]
5086                      <1>      ;jmp     word [SI+D1]
5087                      <1>
5088 00005221 FFA6[7A500000] <1>      jmp     dword [esi+D1]
5089                      <1>
5090                      <1>      ; 07/08/2022
5091                      <1>      ; 13/07/2022
5092                      <1>      BAD_COMMAND:
5093 00005227 C605[37770100]01 <1>      mov     byte [DISK_STATUS1], BAD_CMD ; COMMAND ERROR
5094 0000522E B000          <1>      mov     al, 0
5095 00005230 C3          <1>      retn
5096                      <1>
5097                      <1>      ; -----
5098                      <1>      ; 48 bit LBA read/write
5099                      <1>      ; -----
5100                      <1>
5101                      <1>      su10: ; 01/12/2023 ; 28 bit LBA r/w
5102 00005231 89C8          <1>      mov     eax, ecx ; LBA address (21/02/2015)
5103                      <1>      ; 13/07/2022
5104 00005233 88D1          <1>      mov     cl, dl ; 14/02/2015
5105 00005235 EB82          <1>      jmp     short lbarw2
5106                      <1>
5107                      <1>      lba_read_write:
5108                      <1>      ; 02/12/2023
5109                      <1>      ; 01/12/2023 - TRDOS 386 v2.0.7 (48 bit LBA rw)
5110 00005237 81F9FFFFFF0F <1>      cmp     ecx, 0FFFFFFFh
5111 0000523D 731B          <1>      jnb     short su6
5112                      <1>      ;movzx    eax, byte [esp] ; ***
5113 0000523F 31C0          <1>      xor     eax, eax
5114 00005241 8A0424      <1>      mov     al, [esp] ; *** ; sector count
5115 00005244 01C8          <1>      add     eax, ecx
5116 00005246 730B          <1>      jnc     short su5
5117 00005248 C605[37770100]17 <1>      mov     byte [DISK_STATUS1], ERR_INV_PARAMETER
5118                      <1>      ;jmp     short su7
5119                      <1>      ; 20/06/2024
5120 0000524F 58          <1>      pop     eax ; ***
5121 00005250 B000          <1>      mov     al, 0
5122 00005252 C3          <1>      retn
5123                      <1>      su5:
5124 00005253 3DFFFFFF0F <1>      cmp     eax, 0FFFFFFFh ; 28 bit limit
5125 00005258 76D7          <1>      jna     short su10
5126                      <1>      su6:
5127                      <1>
5128                      <1>      ; 20/06/2024 (TRDOS 386 v2.0.8)
5129                      <1>      %if 0
5130                      <1>      ; 48 bit LBA r/w
5131                      <1>      mov     al, [hf_m_s] ; (+!+) ; 02/12/2023
5132                      <1>      shl     al, 4
5133                      <1>      ; 02/12/2023
5134                      <1>      ;add     al, 0E0h
5135                      <1>      add     al, 40h
5136                      <1>      ;;add     al, 0Eh
5137                      <1>      ;add     al, 04h
5138                      <1>      ;shl     al, 4
5139                      <1>      mov     dx, [HF_PORT]
5140                      <1>      add     dl, 6 ; hd base port + 6
5141                      <1>      out     dx, al
5142                      <1>      ; 02/12/2023
5143                      <1>      mov     [CMD_BLOCK+6], al
5144                      <1>      inc     edx ; hd base port + 7
5145                      <1>      in     al, dx
5146                      <1>      NEWIODELAY
5147                      <1>      ;test    al, 128 ; READY ?
5148                      <1>      and     al, 128
5149                      <1>      jz     short su8
5150                      <1>      in     al, dx
5151                      <1>      ;test    al, 128
5152                      <1>      and     al, 128
5153                      <1>      jz     short su8
5154                      <1>      mov     byte [DISK_STATUS1], TIME_OUT
5155                      <1>      su7:
5156                      <1>      pop     eax ; ***
5157                      <1>      mov     al, 0
5158                      <1>      retn
5159                      <1>      su8:
5160                      <1>      ;mov     dx, [HF_PORT]
5161                      <1>      ;inc     edx
5162                      <1>      ;inc     edx ; hd base port + 2
5163                      <1>      sub     dl, 5 ; hd base port + 2
5164                      <1>      ;xor     al, al
5165                      <1>      out     dx, al ; sector count hb (bits 8 to 15) = 0
5166                      <1>      inc     edx ; hd base port + 3
5167                      <1>      mov     eax, ecx ; LBA disk sector address
5168                      <1>      rol     eax, 8
5169                      <1>      out     dx, al ; LBA byte 4 (bits 24 to 31)
5170                      <1>      inc     edx ; hd base port + 4
5171                      <1>      xor     al, al
5172                      <1>      out     dx, al ; LBA byte 5 (bits 32 to 39) = 0
5173                      <1>      inc     edx ; hd base port + 5
5174                      <1>      ;sub     al, al
5175                      <1>      out     dx, al ; LBA byte 6 (bits 40 to 47) = 0
5176                      <1>
5177                      <1>      mov     al, [esp] ; ***
5178                      <1>      ; 02/12/2023
5179                      <1>      mov     [CMD_BLOCK+1], al

```

```

5180 <1>
5181 <1> sub dl, 3 ; hd base port + 2
5182 <1> out dx, al ; sector count 1b (bits 0 to 7)
5183 <1> inc edx ; hd base port + 3
5184 <1> mov eax, ecx ; LBA disk sector address
5185 <1> out dx, al ; LBA byte 1 (bits 0 to 7)
5186 <1> inc edx ; hd base port + 4
5187 <1> shr eax, 8
5188 <1> out dx, al ; LBA byte 2 (bits 8 to 15)
5189 <1> inc edx ; hd base port + 5
5190 <1> shr eax, 8
5191 <1> out dx, al ; LBA byte 3 (bits 16 to 23)
5192 <1>
5193 <1> inc edx ; hd base port + 6
5194 <1>
5195 <1> ; 02/12/2023 (not necessary) (!+)
5196 <1> ;mov al, [hf_m_s]
5197 <1> ;shl al, 4
5198 <1> ;add al, 40h
5199 <1> ;out dx, al
5200 <1>
5201 <1> pop eax ; ***
5202 <1>
5203 <1> inc edx ; dx = hd base port + 7
5204 <1> ; command/status port
5205 <1> ;xchg esi, ebx
5206 <1> ;mov edi, ebx
5207 <1> mov edi, esi ; sector buffer
5208 <1> cmp ah, 1Ch
5209 <1> jne short su9
5210 <1> mov al, 34h ; WRITE SECTOR(S) EXT
5211 <1> out dx, al
5212 <1> jmp CMD_WX
5213 <1> su9:
5214 <1> mov al, 24h ; READ SECTOR(S) EXT
5215 <1> out dx, al
5216 <1> jmp CMD_RX
5217 <1> %else
5218 <1> ; 20/06/2024
5219 <1> pop eax ; ***
5220 <1> mov byte [CMD_BLOCK], 0
5221 <1> mov [CMD_BLOCK+1], al ; sector count to r/w
5222 <1> mov [CMD_BLOCK+2], ecx ; LBA disk sector address
5223 <1>
5224 <1> ;;;
5225 <1> ; 23/06/2024
5226 <1> mov byte [LBAMode], 0FFh
5227 <1>
5228 <1> mov al, [hf_m_s]
5229 <1> shl al, 4
5230 <1> ;add al, 40h
5231 <1> or al, 40h
5232 <1> ; al = 40h (for master), 50h (for slave)
5233 <1> mov [hf_m_s], al
5234 <1> ;;;
5235 <1>
5236 <1> cmp ah, 1Ch
5237 <1> jne short su7
5238 <1> ; AH = 1Ch ; TRDOS 386 v2 - INT 33h - LBA WRITE
5239 <1> ; esi = sector buffer
5240 <1> ; 48 bit LBA write
5241 <1> mov byte [CMD_BLOCK+6], 34h ; WRITE SECTOR(S) EXT
5242 <1> jmp CMD_WX
5243 <1> su7:
5244 <1> ; AH = 1Bh ; TRDOS 386 v2 - INT 33h - LBA READ
5245 <1> mov edi, esi ; sector buffer
5246 <1> ; 48 bit LBA read
5247 <1> mov byte [CMD_BLOCK+6], 24h ; READ SECTOR(S) EXT
5248 <1> jmp CMD_RX
5249 <1>
5250 <1> %endif
5251 <1>
5252 <1> ; -----
5253 <1>
5254 <1> ; 09/08/2022
5255 <1> ; 07/08/2022
5256 <1> ; 17/07/2022
5257 <1> ; 16/07/2022 - TRDOS 386 v2.0.5
5258 <1> ; 10/07/2022 - Retro UNIX 386 v1.1 (kernel v0.2.1.5)
5259 <1>
5260 <1> ; -----
5261 <1> ; RESET THE DISK SYSTEM (AH=00H) :
5262 <1> ; -----
5263 <1>
5264 <1> ; 18-1-2015 : one controller reset (not other one)
5265 <1>
5266 <1> DISK_RESET:
5267 <1> cli
5268 <1> in al, INTB01 ; GET THE MASK REGISTER
5269 <1> ;jmp $+2
5270 <1> IODELAY
5271 <2> jmp short $+2
5272 <2> jmp short $+2
5273 <1> ;and al, 0BFh ; ENABLE FIXED DISK INTERRUPT
5274 <1> and al, 3Fh ; 22/12/2014 (IRQ 14 & IRQ 15)
5275 <1> out INTB01, al
5276 <1> sti ; START INTERRUPTS
5277 <1> ; 14/02/2015
5278 <1> ;mov di, dx
5279 <1> ; 24/12/2021
5280 <1> mov edi, edx
5281 <1> ; 04/01/2015
5282 <1> ;xor di, di
5283 <1> drst0:
5284 <1> mov al, 04h ; bit 2 - SRST
5285 <1> ;mov dx, HF_REG_PORT
5286 <1> mov dx, [HF_REG_PORT]
5287 <1> out dx, al ; RESET
5288 <1> ;mov cx, 10 ; DELAY COUNT
5289 <1> ;DRD: dec cx
5290 <1> ; jnz short DRD ; WAIT 4.8 MICRO-SEC
5291 <1> ;mov cx, 2 ; wait for 30 micro seconds
5292 <1> ;mov ecx, 2 ; 21/02/2015
5293 <1> ; 10/07/2022
5294 <1> sub ecx, ecx
5295 <1> mov cl, 2
5296 <1> call WAITF ; (Award Bios 1999 - WAIT_REFRESH,
5297 <1> ; 40 micro seconds)
5298 <1> mov al, [CONTROL_BYTE]
5299 <1> and al, 0Fh ; SET HEAD OPTION
5300 <1> out dx, al ; TURN RESET OFF
5301 <1> call NOT_BUSY
5302 <1> jnz short DRERR ; TIME OUT ON RESET
5303 <1> mov dx, [HF_PORT]

```

```

5302 000052C8 FEC2      <1>      inc     dl ; HF_PORT+1
5303                   <1>      ; 02/01/2015 - Award BIOS 1999 - AHDSK.ASM
5304                   <1>      ;mov     cl, 10
5305                   <1>      ;mov     ecx, 10 ; 21/02/2015
5306                   <1>      ; 17/07/2022
5307                   <1>      ;xor     ecx, ecx
5308 000052CA B10A      <1>      mov     cl, 10
5309                   <1>      drst1:
5310                   <1>      in      al, dx ; GET RESET STATUS
5311 000052CD 3C01      <1>      cmp     al, 1
5312                   <1>      ; 04/01/2015
5313 000052CF 740C      <1>      jz     short drst2
5314                   <1>      ;jnz     short DRERR ; BAD RESET STATUS
5315                   <1>      ; Drive/Head Register - bit 4
5316                   <1>      ;loop    drst1
5317                   <1>      ; 17/07/2022
5318 000052D1 FEC9      <1>      dec     cl
5319 000052D3 75F7      <1>      jnz     short drst1
5320                   <1>      DRERR:
5321 000052D5 C605[37770100]05 <1>      mov     byte [DISK_STATUS1], BAD_RESET ; CARD FAILED
5322 000052DC C3        <1>      retn
5323                   <1>      drst2:
5324                   <1>      ; 14/02/2015
5325                   <1>      ;mov     dx, di
5326                   <1>      ; 07/08/2022
5327 000052DD 89FA      <1>      mov     edx, edi
5328                   <1>      ;drst3:
5329                   <1>      ; 05/01/2015
5330                   <1>      ; shl     di, 1
5331                   <1>      ; 04/01/2015
5332                   <1>      ; mov     ax, [di+hd_cports]
5333                   <1>      ; cmp     ax, [HF_REG_PORT]
5334                   <1>      ; je      short drst4
5335                   <1>      ; mov     [HF_REG_PORT], ax
5336                   <1>      ; 03/01/2015
5337                   <1>      ; mov     ax, [di+hd_ports]
5338                   <1>      ; mov     [HF_PORT], ax
5339                   <1>      ; 05/01/2014
5340                   <1>      ; shr     di, 1
5341                   <1>      ; 04/01/2015
5342                   <1>      ; jmp     short drst0 ; reset other controller
5343                   <1>      ;drst4:
5344                   <1>      ; 05/01/2015
5345                   <1>      ; shr     di, 1
5346                   <1>      ; mov     al, [di+hd_dregs]
5347                   <1>      ; and     al, 10h ; bit 4 only
5348                   <1>      ; shr     al, 4 ; bit 4 -> bit 0
5349                   <1>      ; mov     [hf_m_s], al ; (0 = master, 1 = slave)
5350                   <1>      ;
5351                   <1>      ; 09/08/2022
5352                   <1>      ; (('INIT_DRV' prodedure sets [CMD_BLOCKS+5] value))
5353                   <1>      ;
5354                   <1>      ; mov     al, [hf_m_s] ; 18/01/2015
5355                   <1>      ; test    al, 1
5356                   <1>      ; jnz     short drst6
5357                   <1>      ; jnz     short drst4
5358                   <1>      ; and     byte [CMD_BLOCK+5], 0EFh ; SET TO DRIVE 0
5359                   <1>      ;drst5:
5360                   <1>      drst3:
5361 000052DF E8F9010000 <1>      call    INIT_DRV ; SET MAX HEADS
5362                   <1>      ;mov     dx, di
5363 000052E4 E8C1020000 <1>      call    HDISK_RECAL ; RECAL TO RESET SEEK SPEED
5364                   <1>      ; 04/01/2014
5365                   <1>      ; inc     di
5366                   <1>      ; mov     dx, di
5367                   <1>      ; cmp     dl, [HF_NUM]
5368                   <1>      ; jb      short drst3
5369                   <1>      ;DRE:
5370 000052E9 C605[37770100]00 <1>      mov     byte [DISK_STATUS1], 0 ; IGNORE ANY SET UP ERRORS
5371 000052F0 C3        <1>      retn
5372                   <1>      ;drst6:
5373                   <1>      drst4: ; Drive/Head Register - bit 4
5374                   <1>      ; or      byte [CMD_BLOCK+5], 010h ; SET TO DRIVE 1
5375                   <1>      ; jmp     short drst5
5376                   <1>      ; jmp     short drst3
5377                   <1>      ;
5378                   <1>      ;-----
5379                   <1>      ; DISK STATUS ROUTINE (AH = 01H) :
5380                   <1>      ;-----
5381                   <1>      ;
5382                   <1>      RETURN_STATUS:
5383 000052F1 A0[37770100] <1>      mov     al, [DISK_STATUS1] ; OBTAIN PREVIOUS STATUS
5384 000052F6 C605[37770100]00 <1>      mov     byte [DISK_STATUS1], 0; RESET STATUS
5385 000052FD C3        <1>      retn
5386                   <1>      ;
5387                   <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
5388                   <1>      ;
5389                   <1>      ;-----
5390                   <1>      ; READ LONG (AH = 0AH) :
5391                   <1>      ;-----
5392                   <1>      ;
5393                   <1>      RD_LONG:
5394                   <1>      ;mov     @CMD_BLOCK+6, READ_CMD OR ECC_MODE
5395 000052FE C645FE22 <1>      mov     byte [CMD_BLOCK+6], READ_CMD + ECC_MODE
5396 00005302 EB04      <1>      jmp     short COMMANDI
5397                   <1>      ;
5398                   <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
5399                   <1>      ;
5400                   <1>      ;-----
5401                   <1>      ; DISK READ ROUTINE (AH = 02H) :
5402                   <1>      ;-----
5403                   <1>      ;
5404                   <1>      DISK_READ:
5405 00005304 C645FE20 <1>      mov     byte [CMD_BLOCK+6], READ_CMD
5406                   <1>      ; jmp     COMMANDI
5407                   <1>      ;
5408                   <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
5409                   <1>      ;
5410                   <1>      ;-----
5411                   <1>      ; COMMANDI :
5412                   <1>      ; REPEATEDLY INPUTS DATA TILL :
5413                   <1>      ; NSECTOR RETURNS ZERO :
5414                   <1>      ;-----
5415                   <1>      COMMANDI:
5416                   <1>      ; 16/07/2022
5417                   <1>      ; (check 64K boundary is not needed)
5418                   <1>      ; call    CHECK_DMA ; CHECK 64K BOUNDARY ERROR
5419                   <1>      ; jc      short CMD_ABORT
5420                   <1>      ;
5421                   <1>      ;mov     di, bx
5422 00005308 89DF      <1>      mov     edi, ebx ; 21/02/2015
5423                   <1>      ;
5424                   <1>      CMD_RX:
5425 0000530A E821030000 <1>      ; 20/06/2024 (48 bit LBA r/w modification)
5425                   <1>      call    COMMAND ; OUTPUT COMMAND

```

```

5426 0000530F 7548      <1>      jnz      short CMD_ABORT
5427                   <1> ;CMD_RX: ; 01/12/2023 (48 bit LBA read)
5428                   <1> CMD_I1:
5429 00005311 E8D4030000 <1>      call     _WAIT          ; WAIT FOR DATA REQUEST INTERRUPT
5430 00005316 7541      <1>      jnz      short TM_OUT          ; TIME OUT
5431                   <1> cmd_ilx:
5432                   <1>      ; 18/02/2016
5433                   <1>      ;;mov     cx, 256          ; SECTOR SIZE IN WORDS
5434                   <1>      ;mov     ecx, 256 ; 21/02/2015
5435                   <1>      ; 16/07/2022
5436 00005318 29C9      <1>      sub     ecx, ecx
5437 0000531A FEC5      <1>      inc     ch ; ecx = 256
5438                   <1>      ;mov     dh, HF_PORT
5439 0000531C 668B15[6E660000] <1>      mov     dx, [HF_PORT]
5440 00005323 FA        <1>      cli
5441 00005324 FC        <1>      cld
5442 00005325 F3666D    <1>      rep     insw          ; GET THE SECTOR
5443 00005328 FB        <1>      sti
5444                   <1>
5445 00005329 F645FE02   <1>      test    byte [CMD_BLOCK+6], ECC_MODE ; CHECK FOR NORMAL INPUT
5446 0000532D 7418      <1>      jz       short CMD_I3
5447 0000532F E80D040000 <1>      call     WAIT_DRQ          ; WAIT FOR DATA REQUEST
5448 00005334 7223      <1>      jc       short TM_OUT
5449                   <1>      ;mov     dx, HF_PORT
5450 00005336 668B15[6E660000] <1>      mov     dx, [HF_PORT]
5451                   <1>      ;;mov     cx, 4          ; GET ECC BYTES
5452                   <1>      ;mov     ecx, 4 ; mov cx, 4
5453                   <1>      ; 16/07/2022
5454 0000533D 31C9      <1>      xor     ecx, ecx
5455 0000533F B104      <1>      mov     cl, 4
5456                   <1> CMD_I2:
5457 00005341 EC        <1>      in      al, dx
5458 00005342 8807      <1>      mov     [edi], al ; 21/02/2015; GO SLOW FOR BOARD
5459 00005344 47        <1>      inc     edi
5460 00005345 E2FA      <1>      loop    CMD_I2
5461                   <1> CMD_I3:
5462                   <1>      ; wait for 400 ns
5463 00005347 80C207     <1>      add     dl, 7
5464 0000534A EC        <1>      in      al, dx
5465 0000534B EC        <1>      in      al, dx
5466 0000534C EC        <1>      in      al, dx
5467                   <1>      ;
5468 0000534D E8E4010000 <1>      call     CHECK_STATUS
5469 00005352 7505      <1>      jnz      short CMD_ABORT          ; ERROR RETURNED
5470 00005354 FE4DF9    <1>      dec     byte [CMD_BLOCK+1] ; CHECK FOR MORE
5471                   <1>      ;jnz      short CMD_I1
5472 00005357 75BF      <1>      jnz      short cmd_ilx ; 18/02/2016
5473                   <1> CMD_ABORT:
5474                   <1> TM_OUT:
5475 00005359 C3        <1>      retn
5476                   <1>
5477                   <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
5478                   <1>
5479                   <1> ;-----
5480                   <1> ; WRITE LONG (AH = 0BH) :
5481                   <1> ;-----
5482                   <1>
5483                   <1> WR_LONG:
5484                   <1>      ;mov     @CMD_BLOCK+6, WRITE_CMD OR ECC_MODE
5485 0000535A C645FE32   <1>      mov     byte [CMD_BLOCK+6], WRITE_CMD + ECC_MODE
5486 0000535E EB04      <1>      jmp     short COMMANDO
5487                   <1>
5488                   <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
5489                   <1>
5490                   <1> ;-----
5491                   <1> ; DISK WRITE ROUTINE (AH = 03H) :
5492                   <1> ;-----
5493                   <1>
5494                   <1> DISK_WRITE:
5495 00005360 C645FE30   <1>      mov     byte [CMD_BLOCK+6], WRITE_CMD
5496                   <1>      ;jmp     COMMANDO
5497                   <1>
5498                   <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
5499                   <1>
5500                   <1> ;-----
5501                   <1> ; COMMANDO :
5502                   <1> ; REPEATEDLY OUTPUTS DATA TILL :
5503                   <1> ; NSECTOR RETURNS ZERO :
5504                   <1> ;-----
5505                   <1> COMMANDO:
5506                   <1>      ; 16/07/2022
5507                   <1>      ; (check 64K boundary is not needed)
5508                   <1>      ;call    CHECK_DMA          ; CHECK 64K BOUNDARY ERROR
5509                   <1>      ;jc       short CMD_ABORT
5510                   <1> CMD_OF:
5511 00005364 89DE      <1>      mov     esi, ebx ; 21/02/2015
5512                   <1> CMD_WX:
5513                   <1>      ; 20/06/2024 (48 bit LBA r/w modification)
5514 00005366 E8C5020000 <1>      call     COMMAND          ; OUTPUT COMMAND
5515 0000536B 75EC      <1>      jnz      short CMD_ABORT
5516                   <1> ;CMD_WX: ; 01/12/2023 (48 bit LBA write)
5517 0000536D E8CF030000 <1>      call     WAIT_DRQ          ; WAIT FOR DATA REQUEST
5518 00005372 72E5      <1>      jc       short TM_OUT          ; TOO LONG
5519                   <1> CMD_O1:
5520                   <1>      ; 16/07/2022
5521 00005374 668B15[6E660000] <1>      mov     dx, [HF_PORT]
5522                   <1>
5523                   <1>      ;mov     ecx, 256 ; 21/02/2015
5524 0000537B 31C9      <1>      xor     ecx, ecx
5525 0000537D FEC5      <1>      inc     ch
5526                   <1>      ; ecx = 256
5527 0000537F FA        <1>      cli
5528 00005380 FC        <1>      cld
5529                   <1>      ;rep     outsw
5530                   <1>      ; 01/12/2023 - TRDOS 386 v2.0.7
5531                   <1> CMD_O1_L:
5532 00005381 666F      <1>      outsw
5533 00005383 EB00      <1>      jmp     $+2
5534 00005385 E2FA      <1>      loop    CMD_O1_L
5535                   <1>
5536 00005387 FB        <1>      sti
5537                   <1>
5538 00005388 F645FE02   <1>      test    byte [CMD_BLOCK+6], ECC_MODE ; CHECK FOR NORMAL OUTPUT
5539 0000538C 7418      <1>      jz       short CMD_O3
5540 0000538E E8AE030000 <1>      call     WAIT_DRQ          ; WAIT FOR DATA REQUEST
5541 00005393 72C4      <1>      jc       short TM_OUT
5542                   <1>      ;mov     dx, HF_PORT
5543 00005395 668B15[6E660000] <1>      mov     dx, [HF_PORT]
5544                   <1>      ; OUTPUT THE ECC BYTES
5545                   <1>      ;mov     ecx, 4 ; mov cx, 4
5546                   <1>      ; 16/07/2022
5547 0000539C 29C9      <1>      sub     ecx, ecx
5548 0000539E B104      <1>      mov     cl, 4
5549                   <1> CMD_O2:

```

```

5550 000053A0 8A06      <1>      mov     al, [esi]
5551 000053A2 EE        <1>      out     dx, al
5552 000053A3 46        <1>      inc     esi
5553 000053A4 E2FA      <1>      loop    CMD_02
5554                                <1>  CMD_03:
5555 000053A6 E83F030000 <1>      call    _WAIT          ; WAIT FOR SECTOR COMPLETE INTERRUPT
5556 000053AB 75AC      <1>      jnz     short TM_OUT      ; ERROR RETURNED
5557 000053AD E884010000 <1>      call    CHECK_STATUS
5558 000053B2 75A5      <1>      jnz     short CMD_ABORT
5559 000053B4 F605[31770100]08 <1>      test    byte [HF_STATUS], ST_DRQ ; CHECK FOR MORE
5560 000053BB 75B7      <1>      jnz     short CMD_01
5561                                <1>      ;mov     dx, HF_PORT+2      ; CHECK RESIDUAL SECTOR COUNT
5562 000053BD 668B15[6E660000] <1>      mov     dx, [HF_PORT]
5563 000053C4 80C202    <1>      add     dl, 2
5564                                <1>      ;inc     dl
5565                                <1>      ;inc     dl
5566 000053C7 EC        <1>      in      al, dx
5567 000053C8 A8FF      <1>      test    al, 0FFh
5568 000053CA 7407      <1>      jz      short CMD_04      ; COUNT = 0 OK
5569 000053CC C605[37770100]BB <1>      mov     byte [DISK_STATUS1], UNDEF_ERR
5570                                <1>      ; OPERATION ABORTED - PARTIAL TRANSFER
5571                                <1>  CMD_04:
5572 000053D3 C3        <1>      retn
5573                                <1>
5574                                <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
5575                                <1>
5576                                <1>  ;-----
5577                                <1>  ;  FORMATTING          (AH = 05H) :
5578                                <1>  ;-----
5579                                <1>
5580                                <1>  FMT_TRK:          ; FORMAT TRACK (AH = 005H)
5581 000053D4 C645FE50    <1>      mov     byte [CMD_BLOCK+6], FMTTRK_CMD
5582                                <1>      ;push    es
5583                                <1>      ;push    bx
5584 000053D8 53        <1>      push    ebx
5585 000053D9 E8F1030000 <1>      call    GET_VEC          ; GET DISK PARAMETERS ADDRESS
5586                                <1>      ;mov     al, [ES:BX+14]      ; GET SECTORS/TRACK
5587 000053DE 8A430E    <1>      mov     al, [ebx+14]
5588 000053E1 8845F9    <1>      mov     [CMD_BLOCK+1], al      ; SET SECTOR COUNT IN COMMAND
5589 000053E4 5B        <1>      pop     ebx
5590                                <1>      ;pop     bx
5591                                <1>      ;pop     es
5592                                <1>      ;jmp     short CMD_OF      ; GO EXECUTE THE COMMAND
5593                                <1>      ; 01/12/2023
5594 000053E5 E97AFFFFFF <1>      jmp     CMD_OF
5595                                <1>
5596                                <1>  ;-----
5597                                <1>  ;  DISK VERIFY          (AH = 04H) :
5598                                <1>  ;-----
5599                                <1>
5600                                <1>  DISK_VERF:
5601 000053EA C645FE40    <1>      mov     byte [CMD_BLOCK+6], VERIFY_CMD
5602 000053EE E83D020000 <1>      call    COMMAND
5603 000053F3 750C      <1>      jnz     short VERF_EXIT      ; CONTROLLER STILL BUSY
5604 000053F5 E8F0020000 <1>      call    _WAIT          ; (Original: CALL WAIT)
5605 000053FA 7505      <1>      jnz     short VERF_EXIT      ; TIME OUT
5606 000053FC E835010000 <1>      call    CHECK_STATUS
5607                                <1>  VERF_EXIT:
5608 00005401 C3        <1>      retn
5609                                <1>
5610                                <1>  ;-----
5611                                <1>  ;  READ DASD TYPE      (AH = 15H) :
5612                                <1>  ;-----
5613                                <1>
5614                                <1>  RETURN_DRIVE_TYPE:
5615                                <1>      ; 10/08/2022
5616                                <1>      ; 13/07/2022
5617                                <1>      ; (Ref: Programmer's Guide to the AMIBIOS -Page 214-, 1992)
5618                                <1>
5619                                <1>      ; INPUT:
5620                                <1>      ;     DL = Disk number (>= 80h)
5621                                <1>      ;     TRDOS 386 v2.0.5 Feature:
5622                                <1>      ;     If AL = 0FFh, return disk size in ECX
5623                                <1>      ;     otherwise in CX:DX
5624                                <1>
5625                                <1>      ; OUTPUT:
5626                                <1>      ;     AH = 00h - No drive present
5627                                <1>      ;     = 03h - Hard disk drive
5628                                <1>      ;     CF = 0 - No error
5629                                <1>      ;     = 1 - Error
5630                                <1>      ;     TRDOS 386 v2.0.5 Feature:
5631                                <1>      ;     AL = 00h - LBA not ready !
5632                                <1>      ;     = 01h - LBA ready
5633                                <1>      ;     (28 bit or 48 bit LBA r/w depending
5634                                <1>      ;     on disk size in CX:DX)
5635                                <1>      ;     CX:DX = Number of 512 byte sectors
5636                                <1>
5637                                <1>      ; (Note: High words of ECX and EDX will be zero at return)
5638                                <1>      ; ((If AL input is 0FFh, disk size will be in ECX only))
5639                                <1>
5640                                <1>  READ_DASD_TYPE:
5641                                <1>  READ_D_T:          ; GET DRIVE PARAMETERS
5642                                <1>      ;push    ds          ; SAVE REGISTERS
5643                                <1>
5644                                <1>      ;push    es
5645                                <1>      ; 18/04/2021
5646                                <1>      ;push    ebx
5647                                <1>      ;call    DDS          ; ESTABLISH ADDRESSING
5648                                <1>      ;push    cs
5649                                <1>      ;pop     ds
5650                                <1>
5651                                <1>      ; 18/04/2021
5652                                <1>      ;mov     bx, KDATA
5653                                <1>      ;mov     ds, bx
5654                                <1>      ;mov     es, bx
5655                                <1>      ;mov     byte [DISK_STATUS1], 0
5656                                <1>      ;mov     b1, [HF_NUM]      ; GET NUMBER OF DRIVES
5657                                <1>      ;and     dl, 7Fh          ; GET DRIVE NUMBER
5658                                <1>      ;cmp     b1, dl
5659                                <1>      ;jbe     short RDT_NOT_PRESENT ; RETURN DRIVE NOT PRESENT
5660                                <1>
5661                                <1>      ;mov     ax, KDATA
5662                                <1>      ;mov     ds, ax
5663 00005402 C605[37770100]00 <1>      mov     byte [DISK_STATUS1], 0
5664 00005409 8A0D[38770100] <1>      mov     c1, [HF_NUM]
5665 0000540F 80E27F    <1>      and     dl, 7Fh
5666 00005412 38D1      <1>      cmp     c1, dl
5667 00005414 7631      <1>      jbe     short RDT_NOT_PRESENT
5668                                <1>
5669                                <1>      ; 18/04/2021 - TRDOS 386 v2.0.4
5670                                <1>
5671                                <1>      ;call    GET_VEC          ; GET DISK PARAMETER ADDRESS
5672                                <1>      ;
5673                                <1>      ;mov     al, [ES:BX+2]      ; HEADS

```



```

5674      <1>      ;mov     al, [ebx+2] ; heads (logical)
5675      <1>      ;;mov    cl, [ES:BX+14]
5676      <1>      ;;mov    cl, [ebx+14]
5677      <1>      ;; 17/04/2021
5678      <1>      ;mov     ah, [ebx+14] ; sectors per track (logical)
5679      <1>      ;;imul   cl           ; * NUMBER OF SECTORS
5680      <1>      ;;mov     cx, [ES:BX]   ; MAX NUMBER OF CYLINDERS
5681      <1>      ;mov     cx, [ebx]      ; cylinders (logical)
5682      <1>      ;; 02/01/2015
5683      <1>      ;; ** leave the last cylinder as reserved for diagnostics **
5684      <1>      ;; (Also in Award BIOS - 1999, AHDSK.ASM, FUN15 -> sub ax, 1)
5685      <1>      ;dec      cx             ; LEAVE ONE FOR DIAGNOSTICS
5686      <1>      ;;imul   cx             ; NUMBER OF SECTORS
5687      <1>      ;; 17/04/2021
5688      <1>      ;mul     ah
5689      <1>      ;; ax = spt*heads
5690      <1>      ;mul     cx
5691      <1>      ;; dx:ax = number of sectors
5692      <1>      ;
5693      <1>      ;mov     cx, dx           ; HIGH ORDER HALF
5694      <1>      ;mov     dx, ax           ; LOW ORDER HALF
5695      <1>      ;
5696      <1>      ; 18/04/2021
5697      <1>      mov     cl, 2
5698      <1>      add     dl, cl ; hd0 = 2
5699      <1>      ;
5700      <1>      ; 13/07/2022
5701      <1>      movzx   edx, dl
5702      <1>      mov     bl, [edx+drv.status]
5703      <1>      and     bl, 1 ; LBA ready bit (bit 0)
5704      <1>      ;
5705      <1>      shl     dl, cl ; * 4
5706      <1>      ;
5707      <1>      ;mov     eax, [edx+drv.size]
5708      <1>      ;mov     dx, ax
5709      <1>      ;shr     eax, 16
5710      <1>      ;mov     cx, ax
5711      <1>      ;
5712      <1>      mov     ecx, [edx+drv.size]
5713      <1>      ;
5714      <1>      ; 13/07/2022
5715      <1>      ;cmp     al, 0FFh ; return disk size in ecx ?
5716      <1>      ;je      short RDT1 ; yes
5717      <1>      ;inc     al ; 0FFh -> 0
5718      <1>      ;jz      short RDT1
5719      <1>      ;
5720      <1>      mov     dx, cx
5721      <1>      shr     ecx, 16
5722      <1>      ;
5723      <1>      ; 13/07/2022
5724      <1>      ; ebx = esp+20
5725      <1>      ; ecx = esp+16
5726      <1>      ; edx = esp+12
5727      <1>      ; esi = esp+8
5728      <1>      ; edi = esp+4
5729      <1>      ;
5730      <1>      ; return disk size in user's registers
5731      <1>      mov     [esp+12], edx
5732      <1>      ; cx:dx = disk size
5733      <1>      RDT1:
5734      <1>      ;mov     [esp+16], ecx
5735      <1>      ;
5736      <1>      ;;sub     al, al
5737      <1>      sub     eax, eax
5738      <1>      mov     ah, 03h ; INDICATE FIXED DISK
5739      <1>      mov     al, bl
5740      <1>      ; al = 1 -> LBA r/w ready/applicable
5741      <1>      ; = 0 -> LBA r/w not ready/applicable
5742      <1>      ; cf = 0
5743      <1>      RDT2:
5744      <1>      mov     [esp+16], ecx
5745      <1>      ;RDT2:
5746      <1>      ; 13/07/2022
5747      <1>      ; 18/04/2021
5748      <1>      ;pop     ebx ; RESTORE REGISTERS
5749      <1>      ;;pop     es
5750      <1>      ;pop     ds
5751      <1>      ; (*) c1c ; CLEAR CARRY
5752      <1>      ;retf    2
5753      <1>      ; (*) 29/05/2016
5754      <1>      ; (*) retf 4
5755      <1>      ;and     byte [esp+8], 0FEh ; clear carry bit of eflags register
5756      <1>      ;iretd
5757      <1>      ;
5758      <1>      ; 13/07/2022
5759      <1>      ; [DISK_STATUS1] = 0
5760      <1>      ; ah = 3
5761      <1>      ; al = 0 or 1 (LBA ready)
5762      <1>      ; cf = 0
5763      <1>      ;
5764      <1>      retn
5765      <1>      ;
5766      <1>      RDT_NOT_PRESENT:
5767      <1>      ;;sub     ax, ax ; DRIVE NOT PRESENT RETURN
5768      <1>      ;; 18/04/2021
5769      <1>      ;sub     eax, eax
5770      <1>      ;;mov     cx, ax ; ZERO BLOCK COUNT
5771      <1>      ;;mov     dx, ax
5772      <1>      ;mov     ecx, eax
5773      <1>      ;mov     edx, eax
5774      <1>      ;jmp     short RDT2
5775      <1>      ; 13/07/2022
5776      <1>      sub     ecx, ecx
5777      <1>      mov     cl, al ; if AL = 0FFh, disk size will be in ECX
5778      <1>      ; if not, disk size will be in CX:DX
5779      <1>      sub     eax, eax
5780      <1>      inc     cl ; 0FFh -> 0
5781      <1>      cmp     cl, 1
5782      <1>      jnb     short RDT2 ; ecx = 0
5783      <1>      ; 10/08/2022
5784      <1>      sub     cl, cl
5785      <1>      ; ecx = 0
5786      <1>      mov     [esp+12], eax ; edx = 0
5787      <1>      stc
5788      <1>      jmp     short RDT2 ; cf = 1, eax = 0
5789      <1>      ;
5790      <1>      ; 10/08/2022
5791      <1>      ; 07/08/2022
5792      <1>      ; 13/07/2022 - TRDOS 386 v2.0.5
5793      <1>      ; 28/05/2016
5794      <1>      ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
5795      <1>      ;
5796      <1>      ;-----
5797      <1>      ; GET PARAMETERS (AH = 08H) :

```

```

5798 <1> ;-----
5799 <1>
5800 <1> GET_PARM_N:
5801 <1> ; ebx = user's buffer address for parameters table
5802 <1> ; 10/08/2022
5803 <1> ; 13/07/2022
5804 <1> ; (if ebx = 0, HDPT will not be returned to user)
5805 <1> ;
5806 <1> ;GET_PARM: ; GET DRIVE PARAMETERS
5807 <1> ;push ds ; SAVE REGISTERS
5808 <1> ;push es
5809 <1>
5810 <1> ;push ebx
5811 0000545D 89DF <1> mov edi, ebx ; 13/07/2022
5812 <1>
5813 <1> ; 13/07/2022
5814 <1> ; ((IBM PC XT-286 ROM BIOS source code remainders))
5815 <1> ;mov ax, ABS0 ; ESTABLISH ADDRESSING
5816 <1> ;mov ds, ax
5817 <1>
5818 <1> ;test dl, 1 ; CHECK FOR DRIVE 1
5819 <1> ;jz short G0
5820 <1> ;les bx, @HF1_TBL_VEC
5821 <1> ;jmp short G1
5822 <1> ;;G0:
5823 <1> ;les bx, @HF_TBL_VEC
5824 <1> ;;G1:
5825 <1> ;call DDS ; ESTABLISH SEGMENT
5826 <1>
5827 <1> ; 13/07/2022
5828 <1> ; 22/12/2014
5829 <1> ;push cs
5830 <1> ;pop ds
5831 <1> ;mov bx, KDATA
5832 <1> ;mov ds, bx
5833 <1> ;mov es, bx ; 27/05/2016
5834 <1> ;
5835 <1> ; 18/04/2021
5836 0000545F 29C9 <1> sub ecx, ecx
5837 <1> ;
5838 00005461 80EA80 <1> sub dl, 80h
5839 00005464 80FA04 <1> cmp dl, MAX_FILE ; TEST WITHIN RANGE
5840 00005467 7344 <1> jae short G2 ; 13/07/2022
5841 <1> ;
5842 <1> ; 21/02/2015
5843 00005469 31DB <1> xor ebx, ebx
5844 <1> ; 18/04/2021
5845 <1> ;sub ecx, ecx
5846 <1> ; 22/12/2014
5847 0000546B 88D3 <1> mov bl, dl
5848 <1> ;xor bh, bh
5849 0000546D C0E302 <1> shl bl, 2 ; convert index to offset
5850 <1> ;add bx, HF_TBL_VEC
5851 00005470 81C3[3C770100] <1> add ebx, HF_TBL_VEC
5852 <1> ;mov ax, [bx+2]
5853 <1> ;mov es, ax ; dpt segment
5854 <1> ;mov bx, [bx] ; dpt offset
5855 00005476 8B1B <1> mov ebx, [ebx] ; 32 bit offset
5856 <1> ; 18/04/2021
5857 00005478 29D2 <1> sub edx, edx
5858 0000547A 8815[37770100] <1> mov [DISK_STATUS1], dl ; 0
5859 <1>
5860 <1> ;mov byte [DISK_STATUS1], 0
5861 <1> ;mov ax, [ES:BX] ; MAX NUMBER OF CYLINDERS
5862 00005480 668B03 <1> mov ax, [ebx]
5863 <1> ;sub ax, 2 ; ADJUST FOR 0-N
5864 00005483 6648 <1> dec ax ; max. cylinder number
5865 00005485 88C5 <1> mov ch, al
5866 00005487 66250003 <1> and ax, 0300h ; HIGH TWO BITS OF CYLINDER
5867 <1> ;shr ax, 1
5868 <1> ;shr ax, 1
5869 <1> ; 13/07/2022
5870 <1> ;shr ax, 2
5871 <1> ; 07/08/2022
5872 0000548B C1E802 <1> shr eax, 2
5873 <1> ;or al, [ES:BX+14] ; SECTORS
5874 0000548E 0A430E <1> or al, [ebx+14]
5875 00005491 88C1 <1> mov cl, al
5876 <1> ;mov dh, [ES:BX+2] ; HEADS
5877 00005493 8A7302 <1> mov dh, [ebx+2]
5878 00005496 FECE <1> dec dh ; 0-N RANGE
5879 00005498 8A15[38770100] <1> mov dl, [HF_NUM] ; DRIVE COUNT
5880 <1> ;sub ax, ax
5881 <1> ; 18/04/2021
5882 <1> ;sub eax, eax
5883 <1>
5884 <1> ; 27/12/2014
5885 <1> ;mov di, bx ; HDPT offset
5886 <1>
5887 <1> ; 13/07/2022
5888 <1> ; ebx = esp+20
5889 <1> ; ecx = esp+16
5890 <1> ; edx = esp+12
5891 <1> ; esi = esp+8
5892 <1> ; edi = esp+4
5893 <1>
5894 <1> ; 13/07/2022
5895 <1> ; set return register contents/values
5896 0000549E 894C2410 <1> mov [esp+16], ecx
5897 000054A2 8954240C <1> mov [esp+12], edx
5898 <1>
5899 <1> ; is hard disk parameters table requested ?
5900 000054A6 09FF <1> or edi, edi ; (edi = [ebp+24] = ebx)
5901 000054A8 751B <1> jnz short G3 ; yes
5902 <1>
5903 000054AA 29C0 <1> sub eax, eax
5904 <1>
5905 <1> ; [DISK_STATUS1] = 0
5906 <1> ; eax = 0
5907 <1> ; cf = 0
5908 <1>
5909 000054AC C3 <1> retn
5910 <1>
5911 <1> G2:
5912 <1> ;mov ah, INIT_FAIL
5913 <1> ;mov byte [DISK_STATUS1], ah ; (INIT_FAIL)
5914 <1> ; ; OPERATION FAILED
5915 <1> ;sub al, al
5916 <1> ;sub dx, dx
5917 <1> ;sub cx, cx
5918 <1> ; 18/04/2021
5919 000054AD 29C0 <1> sub eax, eax
5920 000054AF B407 <1> mov ah, INIT_FAIL
5921 000054B1 8825[37770100] <1> mov [DISK_STATUS1], ah ; OPERATION FAILED

```

```

5922 <1>
5923 <1> ; 13/07/2022
5924 <1> ;sub edx, edx
5925 <1> ;sub ecx, ecx
5926 <1> ; ecx = 0
5927 000054B7 894C240C <1> mov [esp+12], ecx ; 0 ; edx (heads-1, drive count)
5928 000054BB 894C2410 <1> mov [esp+16], ecx ; 0 ; ecx (cylinders-1, sectors)
5929 000054BF 894C2414 <1> mov [esp+20], ecx ; 0 ; ebx (HDPT address)
5930 <1>
5931 000054C3 F9 <1> stc
5932 000054C4 C3 <1> retn
5933 <1>
5934 <1> ; 13/07/2022
5935 <1> ; 29/05/2016 (*)
5936 <1> ; ;stc ; SET ERROR FLAG
5937 <1> ; ;jmp short G5
5938 <1> ; ;jmp short _G6
5939 <1>
5940 <1> G3:
5941 <1> ; 27/05/2016
5942 <1> ; return fixed disk parameters table to user
5943 <1> ; in user's buffer, which is pointed by EBX
5944 <1>
5945 <1> ;xchg edi, [esp] ; ebx (input)-> edi, edi -> [esp]
5946 <1> ; 13/07/2022
5947 <1> ;pop edi
5948 <1> ; edi = user's buffer address
5949 <1> ;push esi
5950 000054C5 89DE <1> mov esi, ebx ; hard disk parameter table (32 bytes)
5951 <1> ;mov ebx, edi ; ebx = user's buffer address
5952 000054C7 51 <1> push ecx
5953 <1> ;push eax
5954 <1> ;mov ecx, 32 ; 32 bytes
5955 000054C8 30ED <1> xor ch, ch
5956 000054CA B120 <1> mov cl, 32
5957 <1> ; ecx = 32
5958 000054CC E83AB60000 <1> call transfer_to_user_buffer ; trdosk6.s (16/05/2016)
5959 <1> ;pop eax
5960 000054D1 59 <1> pop ecx
5961 <1> ; 10/08/2022
5962 <1> ;pop esi
5963 <1> ;pop edi
5964 000054D2 7306 <1> jnc short G4
5965 <1> ; 29/05/2016 (*)
5966 000054D4 B8FF000000 <1> mov eax, 0FFh ; unknown error !
5967 <1> ; [DISK_STATUS1] = 0
5968 <1> ; ah = 0, al = 0FFh
5969 <1> ; cf = 1
5970 <1>
5971 000054D9 C3 <1> retn
5972 <1> ;_G6:
5973 <1> ;or byte [esp+16], 1 ; set carry bit of eflags register
5974 <1> ;G5:
5975 <1> ; 27/05/2016
5976 <1> ;pop ebx ; RESTORE REGISTERS
5977 <1> ;pop es
5978 <1> ;pop ds
5979 <1> ;retf 2
5980 <1> ; (*) 29/05/2016
5981 <1> ; (*) retf 4
5982 <1> ; (*) or byte [esp+8], 1 ; set carry bit of eflags register
5983 <1> ;iretd
5984 <1>
5985 <1> G4:
5986 <1> ; 13/07/2022
5987 000054DA 31C0 <1> xor eax, eax
5988 <1>
5989 <1> ; [user_buffer] = [ebp+24] = HDPT
5990 <1> ; [DISK_STATUS1] = 0
5991 <1> ; eax = 0
5992 <1> ; cf = 0
5993 <1>
5994 000054DC C3 <1> retn
5995 <1>
5996 <1> ; 16/07/2022 - TRDOS 386 v2.0.5
5997 <1>
5998 <1> ;-----
5999 <1> ; INITIALIZE DRIVE (AH = 09H) :
6000 <1> ;-----
6001 <1> ; 03/01/2015
6002 <1> ; According to ATA-ATAPI specification v2.0 to v5.0
6003 <1> ; logical sector per logical track
6004 <1> ; and logical heads - 1 would be set but
6005 <1> ; it is seen as it will be good
6006 <1> ; if physical parameters will be set here
6007 <1> ; because, number of heads <= 16.
6008 <1> ; (logical heads usually more than 16)
6009 <1> ; NOTE: ATA logical parameters (software C, H, S)
6010 <1> ; == INT 13h physical parameters
6011 <1>
6012 <1> ;INIT_DRV:
6013 <1> ; mov byte [CMD_BLOCK+6], SET_PARM_CMD
6014 <1> ; call GET_VEC ; ES:BX -> PARAMETER BLOCK
6015 <1> ; mov al, [es:bx+2] ; GET NUMBER OF HEADS
6016 <1> ; dec al ; CONVERT TO 0-INDEX
6017 <1> ; mov ah, [CMD_BLOCK+5] ; GET SDH REGISTER
6018 <1> ; and ah, 0F0h ; CHANGE HEAD NUMBER
6019 <1> ; or ah, al ; TO MAX HEAD
6020 <1> ; mov [CMD_BLOCK+5], ah
6021 <1> ; mov al, [es:bx+14] ; MAX SECTOR NUMBER
6022 <1> ; mov [CMD_BLOCK+1], al
6023 <1> ; sub ax, ax
6024 <1> ; mov [CMD_BLOCK+3], al ; ZERO FLAGS
6025 <1> ; call COMMAND ; TELL CONTROLLER
6026 <1> ; jnz short INIT_EXIT ; CONTROLLER BUSY ERROR
6027 <1> ; call NOT_BUSY ; WAIT FOR IT TO BE DONE
6028 <1> ; jnz short INIT_EXIT ; TIME OUT
6029 <1> ; call CHECK_STATUS
6030 <1> ;INIT_EXIT:
6031 <1> ; retn
6032 <1>
6033 <1> ; 16/07/2022 - TRDOS 386 v2.0.5
6034 <1>
6035 <1> ; 04/01/2015
6036 <1> ; 02/01/2015 - Derived from from AWARD BIOS 1999
6037 <1> ; AHDSK.ASM - INIT_DRIVE
6038 <1>
6039 <1> INIT_DRV:
6040 000054DD 31C0 <1> ;xor ah,ah
6041 000054DF B00B <1> xor eax, eax ; 21/02/2015
6042 000054E1 3825[4c770100] <1> mov al, 11 ; Physical heads from translated HDPT
6043 000054E7 7702 <1> cmp [LBAMode], ah ; 0
6044 000054E9 B002 <1> ja short idrv0
6045 <1> mov al, 2 ; Physical heads from standard HDPT
6045 <1> idrv0:

```

```

6046      <1>      ; DL = drive number (0 based)
6047 000054EB E8DF020000 <1>      call GET_VEC
6048      <1>      ;push bx
6049 000054F0 53 <1>      push ebx ; 21/02/2015
6050      <1>      ;add bx, ax
6051 000054F1 01C3 <1>      add ebx, eax
6052      <1>      ;; 05/01/2015
6053 000054F3 8A25[72660000] <1>      mov ah, [hf_m_s] ; drive number (0= master, 1= slave)
6054      <1>      ;;and ah, 1
6055 000054F9 C0E404 <1>      shl ah, 4
6056 000054FC 80CCA0 <1>      or ah, 0A0h ; Drive/Head register - 10100000b (A0h)
6057      <1>      ;mov al, [es:bx]
6058 000054FF 8A03 <1>      mov al, [ebx] ; 21/02/2015
6059 00005501 FEC8 <1>      dec al ; last head number
6060      <1>      ;and al, 0Fh
6061 00005503 08E0 <1>      or al, ah ; lower 4 bits for head number
6062      <1>      ;
6063 00005505 C645FE91 <1>      mov byte [CMD_BLOCK+6], SET_PARM_CMD
6064 00005509 8845FD <1>      mov [CMD_BLOCK+5], al
6065      <1>      ;pop bx
6066 0000550C 5B <1>      pop ebx
6067 0000550D 29C0 <1>      sub eax, eax ; 21/02/2015
6068 0000550F B004 <1>      mov al, 4 ; Physical sec per track from translated HDPT
6069 00005511 803D[4c770100]00 <1>      cmp byte [LBAMode], 0
6070 00005518 7702 <1>      ja short idrv1
6071 0000551A B00E <1>      mov al, 14 ; Physical sec per track from standard HDPT
6072      <1>      idrv1:
6073      <1>      ;xor ah, ah
6074      <1>      ;add bx, ax
6075 0000551C 01C3 <1>      add ebx, eax ; 21/02/2015
6076      <1>      ;mov al, [es:bx]
6077      <1>      ; sector number
6078 0000551E 8A03 <1>      mov al, [ebx]
6079 00005520 8845F9 <1>      mov [CMD_BLOCK+1], al
6080 00005523 28C0 <1>      sub al, al
6081 00005525 8845FB <1>      mov [CMD_BLOCK+3], al ; ZERO FLAGS
6082 00005528 E803010000 <1>      call COMMAND ; TELL CONTROLLER
6083 0000552D 751E <1>      jnz short INIT_EXIT ; CONTROLLER BUSY ERROR
6084 0000552F E8E6010000 <1>      call NOT_BUSY ; WAIT FOR IT TO BE DONE
6085 00005534 7517 <1>      jnz short INIT_EXIT ; TIME OUT
6086      <1>      ; 16/07/2022
6087      <1>      ;call CHECK_STATUS
6088      <1>      ;jmp short CHECK_STATUS
6089      <1>      ;INIT_EXIT:
6090      <1>      ;retn
6091      <1>
6092      <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
6093      <1>      ; 10/07/2022 - Retro UNIX 386 v1.1 (kernel v0.2.1.5)
6094      <1>
6095      <1>      ;-----
6096      <1>      ; CHECK FIXED DISK STATUS :
6097      <1>      ;-----
6098      <1>      CHECK_STATUS:
6099 00005536 E831020000 <1>      call CHECK_ST ; CHECK THE STATUS BYTE
6100      <1>      ;jnz short CHECK_S1 ; AN ERROR WAS FOUND
6101      <1>      ; 10/07/2022
6102 0000553B 7510 <1>      jnz short CHECK_S2
6103 0000553D A801 <1>      test al, ST_ERROR ; WERE THERE ANY OTHER ERRORS
6104 0000553F 7405 <1>      jz short CHECK_S1 ; NO ERROR REPORTED
6105 00005541 E866020000 <1>      call CHECK_ER ; ERROR REPORTED
6106      <1>      CHECK_S1:
6107 00005546 803D[37770100]00 <1>      cmp byte [DISK_STATUS1], 0 ; SET STATUS FOR CALLER
6108      <1>      CHECK_S2:
6109      <1>      INIT_EXIT: ; 10/07/2022
6110 0000554D C3 <1>      ret
6111      <1>
6112      <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
6113      <1>
6114      <1>      ;-----
6115      <1>      ; SEEK (AH = 0CH) :
6116      <1>      ;-----
6117      <1>
6118      <1>      DISK_SEEK:
6119 0000554E C645FE70 <1>      mov byte [CMD_BLOCK+6], SEEK_CMD
6120 00005552 E8D9000000 <1>      call COMMAND
6121 00005557 751C <1>      jnz short DS_EXIT ; CONTROLLER BUSY ERROR
6122 00005559 E88C010000 <1>      call _WAIT
6123 0000555E 7515 <1>      jnz short DS_EXIT ; TIME OUT ON SEEK
6124 00005560 E8D1FFFFF <1>      call CHECK_STATUS
6125 00005565 803D[37770100]40 <1>      cmp byte [DISK_STATUS1], BAD_SEEK
6126 0000556C 7507 <1>      jne short DS_EXIT
6127 0000556E C605[37770100]00 <1>      mov byte [DISK_STATUS1], 0
6128      <1>      DS_EXIT:
6129 00005575 C3 <1>      ret
6130      <1>
6131      <1>      ;-----
6132      <1>      ; TEST DISK READY (AH = 10H) :
6133      <1>      ;-----
6134      <1>
6135      <1>      TST_RDY:
6136 00005576 E89F010000 <1>      call NOT_BUSY ; WAIT FOR CONTROLLER
6137 0000557B 752C <1>      jnz short TR_EX
6138      <1>      ;;;
6139      <1>      ; 23/06/2024
6140 0000557D 803D[4c770100]FF <1>      cmp byte [LBAMode], 0FFh
6141 00005584 7507 <1>      jne short tst_28bit_rdy
6142 00005586 A0[72660000] <1>      mov al, [hf_m_s]
6143 0000558B E803 <1>      jmp short tst_48bit_rdy
6144      <1>      tst_28bit_rdy:
6145      <1>      ;;;
6146 0000558D 8A45FD <1>      mov al, [CMD_BLOCK+5] ; SELECT DRIVE
6147      <1>      tst_48bit_rdy:
6148 00005590 668B15[6E660000] <1>      mov dx, [HF_PORT]
6149 00005597 80C206 <1>      add dl, 6
6150 0000559A EE <1>      out dx, al
6151 0000559B E8CC010000 <1>      call CHECK_ST ; CHECK STATUS ONLY
6152 000055A0 7507 <1>      jnz short TR_EX
6153 000055A2 C605[37770100]00 <1>      mov byte [DISK_STATUS1], 0 ; WIPE OUT DATA CORRECTED ERROR
6154      <1>      TR_EX:
6155 000055A9 C3 <1>      ret
6156      <1>
6157      <1>      ;-----
6158      <1>      ; RECALIBRATE (AH = 11H) :
6159      <1>      ;-----
6160      <1>
6161      <1>      HDISK_RECAL:
6162 000055AA C645FE10 <1>      mov byte [CMD_BLOCK+6], RECAL_CMD ; 10h, 16
6163 000055AE E87D000000 <1>      call COMMAND ; START THE OPERATION
6164 000055B3 7523 <1>      jnz short RECAL_EXIT ; ERROR
6165 000055B5 E830010000 <1>      call _WAIT ; WAIT FOR COMPLETION
6166 000055BA 7407 <1>      jz short RECAL_X ; TIME OUT ONE OK ?
6167 000055BC E829010000 <1>      call _WAIT ; WAIT FOR COMPLETION LONGER
6168 000055C1 7515 <1>      jnz short RECAL_EXIT ; TIME OUT TWO TIMES IS ERROR
6169      <1>      RECAL_X:

```

```

6170 000055C3 E86FFFFFFF <1> call CHECK_STATUS
6171 000055C8 803D[37770100]40 <1> cmp byte [DISK_STATUS1], BAD_SEEK ; SEEK NOT COMPLETE
6172 000055CF 7507 <1> jne short RECAL_EXIT ; IS OK
6173 000055D1 C605[37770100]00 <1> mov byte [DISK_STATUS1], 0
6174 <1> RECAL_EXIT:
6175 000055D8 803D[37770100]00 <1> cmp byte [DISK_STATUS1], 0
6176 000055DF C3 <1> retn
6177 <1>
6178 <1> ;-----
6179 <1> ; CONTROLLER DIAGNOSTIC (AH = 14H) :
6180 <1> ;-----
6181 <1>
6182 <1> CTLR_DIAGNOSTIC:
6183 <1> ; 07/08/2022 - TRDOS 386 v2.0.5
6184 000055E0 FA <1> cli ; DISABLE INTERRUPTS WHILE CHANGING MASK
6185 000055E1 E4A1 <1> in al, INTB01 ; TURN ON SECOND INTERRUPT CHIP
6186 <1> ;and al, 0BFH
6187 000055E3 243F <1> and al, 3Fh ; enable IRQ 14 & IRQ 15
6188 <1> ;jmp $+2
6189 <1> IODELAY
6190 <2> jmp short $+2
6191 <2> jmp short $+2
6192 000055E5 EB00 <1> out INTB01, al
6193 000055E7 EB00 <1> IODELAY
6194 000055E9 E6A1 <2> jmp short $+2
6195 <2> jmp short $+2
6196 000055EB EB00 <1> in al, INTA01 ; LET INTERRUPTS PASS THRU TO
6197 000055ED EB00 <1> and al, 0FBh ; SECOND CHIP
6198 000055EF E421 <1> ;jmp $+2
6199 000055F1 24FB <1> IODELAY
6200 <2> jmp short $+2
6201 <2> jmp short $+2
6202 000055F3 EB00 <1> out INTA01, al
6203 000055F5 EB00 <1> sti
6204 000055F7 E621 <1> call NOT_BUSY ; WAIT FOR CARD
6205 000055F9 FB <1> jnz short CD_ERR ; BAD CARD
6206 000055FA E81B010000 <1> ;mov dx, PORT+7
6207 000055FF 7526 <1> mov dx, [HF_PORT]
6208 <1> add dl, 7
6209 00005601 668B15[6E660000] <1> mov al, DIAG_CMD ; START DIAGNOSE
6210 00005608 80C207 <1> out dx, al
6211 0000560B B090 <1> call NOT_BUSY ; WAIT FOR IT TO COMPLETE
6212 0000560D EE <1> mov ah, TIME_OUT
6213 0000560E E807010000 <1> jnz short CD_EXIT ; TIME OUT ON DIAGNOSTIC
6214 00005613 B480 <1> ;mov dx, HF_PORT+1 ; GET ERROR REGISTER
6215 00005615 7512 <1> mov dx, [HF_PORT]
6216 00005617 668B15[6E660000] <1> inc dl
6217 0000561E FEC2 <1> in al, dx
6218 00005620 EC <1> ; 07/08/2022
6219 <1> ;mov [HF_ERROR], al ; SAVE IT
6220 <1> mov ah, 0
6221 <1> cmp al, 1 ; CHECK FOR ALL OK
6222 <1> je short CD_EXIT
6223 <1> CD_ERR:
6224 <1> mov ah, BAD_CNTLR
6225 <1> CD_EXIT:
6226 <1> mov [DISK_STATUS1], ah
6227 <1> retn
6228 <1>
6229 <1> ; 20/06/2024 - TRDOS 386 v2.0.8
6230 <1> ; 16/07/2022 - TRDOS 386 v2.0.5
6231 <1> ; 10/07/2022 - Retro UNIX 386 v1.1 (Kernel v0.2.1.5)
6232 <1>
6233 <1> ;-----
6234 <1> ; COMMAND :
6235 <1> ; THIS ROUTINE OUTPUTS THE COMMAND BLOCK :
6236 <1> ; OUTPUT :
6237 <1> ; BL = STATUS :
6238 <1> ; BH = ERROR REGISTER :
6239 <1> ;-----
6240 <1> COMMAND:
6241 <1> ;push ebx ; 10/07/2022 ; WAIT FOR SEEK COMPLETE AND READY
6242 <1> ;mov ecx, DELAY_2 ; SET INITIAL DELAY BEFORE TEST
6243 <1> COMMAND1:
6244 <1> ;push ecx ; SAVE LOOP COUNT
6245 <1> call TST_RDY ; CHECK DRIVE READY
6246 <1> ;pop ecx
6247 <1> ;pop ebx ; 10/07/2022
6248 <1> jz short COMMAND2 ; DRIVE IS READY
6249 <1> cmp byte [DISK_STATUS1], TIME_OUT ; TST_RDY TIMED OUT--GIVE UP
6250 <1> ;jz short CMD_TIMEOUT
6251 <1> ;loop COMMAND1 ; KEEP TRYING FOR A WHILE
6252 <1> ;jmp short COMMAND4 ; ITS NOT GOING TO GET READY
6253 <1> jne short COMMAND4
6254 <1> CMD_TIMEOUT:
6255 <1> mov byte [DISK_STATUS1], BAD_CNTLR
6256 <1> COMMAND4:
6257 <1> ;pop ebx ; 10/07/2022
6258 <1> cmp byte [DISK_STATUS1], 0 ; SET CONDITION CODE FOR CALLER
6259 <1> retn
6260 <1> COMMAND2:
6261 <1> ;pop ebx ; 10/07/2022
6262 <1> ;push edi ; 10/07/2022
6263 <1> mov byte [HF_INT_FLAG], 0 ; RESET INTERRUPT FLAG
6264 <1> cli ; INHIBIT INTERRUPTS WHILE CHANGING MASK
6265 <1> in al, INTB01 ; TURN ON SECOND INTERRUPT CHIP
6266 <1> ;and al, 0BFh
6267 <1> and al, 3Fh ; Enable IRQ 14 & 15
6268 <1> ;jmp $+2
6269 <1> IODELAY
6270 <2> jmp short $+2
6271 <2> jmp short $+2
6272 <1> out INTB01, al
6273 <1> in al, INTA01 ; LET INTERRUPTS PASS THRU TO
6274 <1> and al, 0FBh ; SECOND CHIP
6275 <1> ;jmp $+2
6276 <1> IODELAY
6277 <2> jmp short $+2
6278 <2> jmp short $+2
6279 <1> out INTA01, al
6280 <1> sti
6281 <1> ;xor edi, edi ; INDEX THE COMMAND TABLE
6282 <1> ; 10/07/2022
6283 <1> xor ecx, ecx
6284 <1>
6285 <1> ;mov dx, HF_PORT+1 ; DISK ADDRESS
6286 <1> mov dx, [HF_PORT]
6287 <1> inc dl
6288 <1>
6289 <1> ; 20/06/2024
6290 <1> ;mov al, [CMD_BLOCK+6]
6291 <1> ;cmp al, 24h ; READ SECTOR(S) EXT
6292 <1> ;je short COMMAND5

```

```

6284      <1>      ;cmp     al, 34h ; WRITE SECTOR(S) EXT
6285      <1>      ;je      short COMMAND5
6286      <1>      ; 23/06/2024
6287 00005677 803D[4C770100]FF <1>      cmp     byte [LBAMode], 0FFh
6288 0000567E 7428 <1>      je      short COMMAND5 ; 48 bit LBA read/write
6289      <1>
6290      <1>      ; 20/06/2024
6291      <1>      ; dx = hd base port + 1 ; 1F1h or 171h
6292      <1>
6293 00005680 F605[39770100]C0 <1>      test    byte [CONTROL_BYTE], 0C0h ; CHECK FOR RETRY SUPPRESSION
6294 00005687 740E <1>      jz      short COMMAND3
6295      <1>      ; 20/06/2024
6296      <1>      ;mov     al, [CMD_BLOCK+6] ; YES-GET OPERATION CODE
6297 00005689 24F0 <1>      and     al, 0F0h ; GET RID OF MODIFIERS
6298 0000568B 3C20 <1>      cmp     al, 20h ; 20H-40H IS READ, WRITE, VERIFY
6299 0000568D 7208 <1>      jb      short COMMAND3
6300 0000568F 3C40 <1>      cmp     al, 40h
6301 00005691 7704 <1>      ja      short COMMAND3
6302 00005693 804DFE01 <1>      or      byte [CMD_BLOCK+6], NO_RETRIES
6303      <1>      ; VALID OPERATION FOR RETRY SUPPRESS
6304      <1>      COMMAND3:
6305      <1>      ;mov     al, [CMD_BLOCK+edi] ; GET THE COMMAND STRING BYTE
6306      <1>      ; 10/07/2022
6307 00005697 8A440DF8 <1>      mov     al, [CMD_BLOCK+ecx]
6308 0000569B EE <1>      out     dx, al ; GIVE IT TO CONTROLLER
6309      <1>      IODELAY
6310      <2>      jmp     short $+2
6311      <2>      jmp     short $+2
6312      <1>      ;inc     edi ; NEXT BYTE IN COMMAND BLOCK
6313      <1>      ; 10/07/2022
6314 000056A0 41 <1>      inc     ecx
6315      <1>      ;inc     dx ; NEXT DISK ADAPTER REGISTER
6316      <1>      inc     edx ; 10/07/2022
6317 000056A2 80F907 <1>      cmp     di, 7 ; 01/01/2015 ; ALL DONE?
6318 000056A5 72F0 <1>      jne     short COMMAND3 ; NO--GO DO NEXT ONE
6319      <1>      cmp     cl, 7 ; 10/07/2022
6320 000056A7 C3 <1>      jnb     short COMMAND3
6321      <1>      ;pop     edi ; 10/07/2022
6322      <1>      retn ; ZERO FLAG IS SET
6323      <1>      ; 20/06/2024
6324      <1>      COMMAND5:
6325      <1>      ; 48 bit LBA r/w
6326 000056A8 8A45F8 <1>      mov     al, [CMD_BLOCK] ; 0
6327 000056AB EE <1>      out     dx, al ; hd base port + 1
6328      <1>      ;
6329 000056AC 80C205 <1>      add     dl, 5 ; hd base port + 6 ; 1F6h or 176h
6330 000056AF A0[72660000] <1>      mov     al, [hf_m_s]
6331      <1>      ; 23/06/2024
6332      <1>      shl     al, 4
6333      <1>      ;add     al, 40h
6334      <1>      ;or      al, 40h
6335      <1>      ; ; al = 40h (for master), 50h (for slave)
6336      <1>      ;
6337 000056B4 EE <1>      out     dx, al
6338 000056B5 80EA04 <1>      sub     dl, 4 ; hd base port + 2 ; 1F2h or 172h
6339 000056B8 30C0 <1>      xor     al, al ; 0
6340 000056BA EE <1>      out     dx, al ; sector count hb (bits 8 to 15) = 0
6341 000056BB 42 <1>      inc     edx ; hd base port + 3
6342 000056BC 8B45FA <1>      mov     eax, [CMD_BLOCK+2]
6343 000056BF C1C008 <1>      rol     eax, 8
6344 000056C2 EE <1>      out     dx, al ; LBA byte 4 (bits 24 to 31)
6345 000056C3 42 <1>      inc     edx ; hd base port + 4
6346 000056C4 30C0 <1>      xor     al, al
6347 000056C6 EE <1>      out     dx, al ; LBA byte 5 (bits 32 to 39) = 0
6348 000056C7 42 <1>      inc     edx ; hd base port + 5
6349 000056C8 28C0 <1>      sub     al, al
6350 000056CA EE <1>      out     dx, al ; LBA byte 6 (bits 40 to 47) = 0
6351 000056CB 8A45F9 <1>      mov     al, [CMD_BLOCK+1] ; sector count
6352 000056CE 80EA03 <1>      sub     dl, 3 ; hd base port + 2
6353 000056D1 EE <1>      out     dx, al ; sector count lb (bits 0 to 7)
6354 000056D2 C1E808 <1>      shr     eax, 8
6355 000056D5 42 <1>      inc     edx ; hd base port + 3 ; 1F3h or 173h
6356 000056D6 EE <1>      out     dx, al ; LBA byte 1 (bits 0 to 7)
6357 000056D7 C1E808 <1>      shr     eax, 8
6358 000056DA 42 <1>      inc     edx ; hd base port + 4
6359 000056DB EE <1>      out     dx, al ; LBA byte 2 (bits 8 to 15)
6360 000056DC C1E808 <1>      shr     eax, 8
6361 000056DF 42 <1>      inc     edx ; hd base port + 5
6362 000056E0 EE <1>      out     dx, al ; LBA byte 3 (bits 16 to 23)
6363 000056E1 42 <1>      inc     edx ; hd base port + 6
6364 000056E2 8A45FE <1>      mov     al, [CMD_BLOCK+6] ; 48 bit read or write command
6365      <1>      ; al = 24h or 34h
6366 000056E5 42 <1>      inc     edx ; hd base port + 7 ; 1F7h or 177h
6367 000056E6 EE <1>      out     dx, al
6368 000056E7 31C0 <1>      xor     eax, eax ; 0
6369 000056E9 C3 <1>      ; zf = 1
6370      <1>      retn
6371      <1>      ;CMD_TIMEOUT:
6372      <1>      ; mov     byte [DISK_STATUS1], BAD_CNTL
6373      <1>      ;COMMAND4:
6374      <1>      ; pop     ebx
6375      <1>      ; cmp     byte [DISK_STATUS1], 0 ; SET CONDITION CODE FOR CALLER
6376      <1>      ; retn
6377      <1>
6378      <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
6379      <1>      ; 10/07/2022 - Retro UNIX 386 v1.1 (Kernel v0.2.1.5)
6380      <1>
6381      <1>      ;-----
6382      <1>      ; WAIT FOR INTERRUPT :
6383      <1>      ;-----
6384      <1>      ;WAIT:
6385      <1>      _WAIT:
6386 000056EA FB <1>      sti ; MAKE SURE INTERRUPTS ARE ON
6387      <1>      ;sub     cx, cx ; SET INITIAL DELAY BEFORE TEST
6388      <1>      ;clc
6389      <1>      ;mov     ax, 9000h ; DEVICE WAIT INTERRUPT
6390      <1>      ;int     15h
6391      <1>      ;jc      short WT2 ; DEVICE TIMED OUT
6392      <1>      ;mov     bl, DELAY_1 ; SET DELAY COUNT
6393      <1>
6394      <1>      ;mov     bl, WAIT_HDU_INT_HI
6395      <1>      ; ; 21/02/2015
6396      <1>      ; ;mov     bl, WAIT_HDU_INT_HI + 1
6397      <1>      ; ;mov     cx, WAIT_HDU_INT_LO
6398 000056EB B915160500 <1>      mov     ecx, WAIT_HDU_INT_LH
6399      <1>      ; (AWARD BIOS -> WAIT_FOR_MEM)
6400      <1>      ;----- WAIT LOOP
6401      <1>
6402      <1>      WT1:
6403      <1>      ;test    byte [HF_INT_FLAG], 80h ; TEST FOR INTERRUPT
6404 000056F0 F605[32770100]C0 <1>      test    byte [HF_INT_FLAG], 0C0h
6405      <1>      ;loopz   WT1

```

```

6406 000056F7 7512      <1>      jnz      short WT3          ; INTERRUPT--LETS GO
6407                   <1>      ;dec      b1
6408                   <1>      ;jnz      short WT1          ; KEEP TRYING FOR A WHILE
6409                   <1>
6410                   <1> WT1_hi:
6411                   <1>      in       al, SYS1 ; 61h (PORT_B)          ; wait for lo to hi
6412 000056F9 E461       <1>      test     al, 10h          ; transition on memory
6413 000056FB A810       <1>      jnz      short WT1_hi          ; refresh.
6414 000056FD 75FA       <1> WT1_lo:
6415 000056FF E461       <1>      in       al, SYS1          ; 061h (PORT_B)
6416 00005701 A810       <1>      test     al, 10h
6417 00005703 74FA       <1>      jz       short WT1_lo
6418 00005705 E2E9       <1>      loop     WT1
6419                   <1>      ;;or      b1, b1
6420                   <1>      ;;jz      short WT2
6421                   <1>      ;;dec     b1
6422                   <1>      ;;jmp     short WT1
6423                   <1>      ;;dec     b1
6424                   <1>      ;jnz      short WT1
6425                   <1> WT2:
6426                   <1>      ; 10/07/2022
6427                   <1>      ;mov     byte [DISK_STATUS1], TIME_OUT ; REPORT TIME OUT ERROR
6428 00005707 B080       <1>      mov      al, TIME_OUT
6429 00005709 EB07       <1>      jmp      short WT4
6430                   <1> WT3:
6431                   <1>      ;mov     byte [DISK_STATUS1], 0
6432                   <1>      ;mov     byte [HF_INT_FLAG], 0
6433 0000570B 28C0       <1>      sub      al, al ; 0
6434 0000570D A2[32770100] <1>      mov     byte [HF_INT_FLAG], al
6435                   <1> WT4:
6436                   <1> NB2:
6437 00005712 A2[37770100] <1>      mov     byte [DISK_STATUS1], al
6438                   <1>
6439                   <1>      ;cmp     byte [DISK_STATUS1], 0          ; SET CONDITION CODE FOR CALLER
6440 00005717 20C0       <1>      and      al, al
6441                   <1>      ; zf = 0 -> time out, zf = 1 -> ok
6442 00005719 C3         <1>      retn
6443                   <1>
6444                   <1>      ; 16/07/2022 - TRDOS 386 v2.0.5
6445                   <1>      ; 10/07/2022 - Retro UNIX 386 v1.1 (Kernel v0.2.1.5)
6446                   <1>
6447                   <1> ;-----
6448                   <1> ; WAIT FOR CONTROLLER NOT BUSY :
6449                   <1> ;-----
6450                   <1> NOT_BUSY:
6451 0000571A FB         <1>      sti             ; MAKE SURE INTERRUPTS ARE ON
6452                   <1>      ;push    ebx
6453                   <1>      ;sub     cx, cx          ; SET INITIAL DELAY BEFORE TEST
6454 0000571B 668B15[6E660000] <1>      mov     dx, [HF_PORT]
6455 00005722 80C207     <1>      add     dl, 7          ; Status port (HF_PORT+7)
6456                   <1>      ;mov     b1, DELAY_1
6457                   <1>
6458                   <1>      ;mov     cx, WAIT_HDU_INT_LO          ; wait for 10 seconds
6459                   <1>      ;;mov     b1, WAIT_HDU_INT_HI          ; 1615h
6460                   <1>      ;mov     b1, WAIT_HDU_INT_HI + 1      ; 05h
6461 00005725 B915160500 <1>      mov     ecx, WAIT_HDU_INT_LH ; 21/02/2015
6462                   <1>
6463                   <1>      ;;mov     byte [wait_count], 0          ; Reset wait counter
6464                   <1> NB1:
6465 0000572A EC         <1>      in       al, dx          ; CHECK STATUS
6466                   <1>      ;test     al, ST_BUSY
6467 0000572B 2480       <1>      and      al, ST_BUSY
6468                   <1>      ;loopnz  NB1
6469 0000572D 74E3       <1>      jz       short NB2 ; al = 0          ; NOT BUSY--LETS GO
6470                   <1>      ;dec     b1
6471                   <1>      ;jnz      short NB1          ; KEEP TRYING FOR A WHILE
6472                   <1>
6473                   <1> NB1_hi:
6474 0000572F E461       <1>      in       al, SYS1          ; wait for hi to lo
6475 00005731 A810       <1>      test     al, 010h          ; transition on memory
6476 00005733 75FA       <1>      jnz      short NB1_hi          ; refresh.
6477                   <1> NB1_lo:
6478 00005735 E461       <1>      in       al, SYS1
6479 00005737 A810       <1>      test     al, 010h
6480 00005739 74FA       <1>      jz       short NB1_lo
6481 0000573B E2ED       <1>      loop     NB1
6482                   <1>      ;dec     b1
6483                   <1>      ;jnz      short NB1
6484                   <1>
6485                   <1>      ;;cmp     byte [wait_count], 182      ; 10 seconds (182 timer ticks)
6486                   <1>      ;;jb      short NB1
6487                   <1>
6488                   <1>      ;mov     byte [DISK_STATUS1], TIME_OUT ; REPORT TIME OUT ERROR
6489                   <1>      ;jmp      short NB3
6490 0000573D B080       <1>      mov      al, TIME_OUT
6491                   <1> ;NB2:
6492 0000573F EBD1       <1>      jmp      short NB2 ; 10/07/2022
6493                   <1>
6494                   <1>      ;mov     byte [DISK_STATUS1], 0
6495                   <1>      ;;NB3:
6496                   <1>      ;pop     ebx
6497                   <1>      ;mov     [DISK_STATUS1], al          ;;; will be set after return
6498                   <1>      ;;cmp     byte [DISK_STATUS1], 0          ; SET CONDITION CODE FOR CALLER
6499                   <1>      ;;or      al, al          ; (zf = 0 --> timeout)
6500                   <1>      ;retn
6501                   <1>
6502                   <1> ;-----
6503                   <1> ; WAIT FOR DATA REQUEST :
6504                   <1> ;-----
6505                   <1> WAIT_DRQ:
6506                   <1>      ;mov     cx, DELAY_3
6507                   <1>      ;mov     dx, HF_PORT+7
6508 00005741 668B15[6E660000] <1>      mov     dx, [HF_PORT]
6509 00005748 80C207     <1>      add     dl, 7
6510                   <1>      ;;mov     b1, WAIT_HDU_DRQ_HI          ; 0
6511                   <1>      ;;mov     cx, WAIT_HDU_DRQ_LO          ; 1000 (30 milli seconds)
6512                   <1>      ; (but it is written as 2000
6513                   <1>      ; micro seconds in ATORGS.ASM file
6514                   <1>      ; of Award Bios - 1999, D1A0622)
6515 0000574B B9E8030000 <1>      mov     ecx, WAIT_HDU_DRQ_LH ; 21/02/2015
6516                   <1> WQ_1:
6517 00005750 EC         <1>      in       al, dx          ; GET STATUS
6518 00005751 A808       <1>      test     al, ST_DRQ          ; WAIT FOR DRQ
6519 00005753 7516       <1>      jnz      short WQ_OK
6520                   <1>      ;loop     WQ_1          ; KEEP TRYING FOR A SHORT WHILE
6521                   <1> WQ_hi:
6522 00005755 E461       <1>      in       al, SYS1          ; wait for hi to lo
6523 00005757 A810       <1>      test     al, 010h          ; transition on memory
6524 00005759 75FA       <1>      jnz      short WQ_hi          ; refresh.
6525                   <1> WQ_lo:
6526 0000575B E461       <1>      in       al, SYS1
6527 0000575D A810       <1>      test     al, 010h
6528 0000575F 74FA       <1>      jz       short WQ_lo
6529 00005761 E2ED       <1>      loop     WQ_1

```

```

6530
6531 00005763 C605[37770100]80
6532 0000576A F9
6533
6534 0000576B C3
6535
6536
6537
6538
6539
6540
6541
6542
6543
6544
6545
6546 0000576C 668B15[6E660000]
6547 00005773 80C207
6548
6549
6550
6551
6552 00005776 EC
6553
6554
6555
6556
83 00005777 E6EB
6557
6558
6559
6560 00005779 A2[31770100]
6561
6562 0000577E 28E4
6563 00005780 A880
6564 00005782 751A
6565 00005784 B4CC
6566 00005786 A820
6567 00005788 7514
6568 0000578A B4AA
6569 0000578C A840
6570 0000578E 740E
6571 00005790 B440
6572 00005792 A810
6573 00005794 7408
6574 00005796 B411
6575 00005798 A804
6576 0000579A 7502
6577
6578 0000579C 30E4
6579
6580 0000579E 8825[37770100]
6581 000057A4 80FC11
6582 000057A7 7402
6583
6584 000057A9 20E4
6585
6586 000057AB C3
6587
6588
6589
6590
6591
6592
6593
6594
6595 000057AC 668B15[6E660000]
6596 000057B3 FEC2
6597 000057B5 EC
6598
6599
6600
6601 000057B6 29C9
6602
6603 000057B8 B108
6604
6605 000057BA D0E0
6606 000057BC 7202
6607 000057BE E2FA
6608
6609
6610
6611 000057C0 81C1[64660000]
6612
6613
6614
6615
6616 000057C6 8A21
6617
6618 000057C8 8825[37770100]
6619
6620
6621
6622
6623 000057CE C3
6624
6625
6626
6627
6628
6629
6630
6631
6632
6633
6634
6635
6636
6637
6638
6639
6640
6641
6642
6643
6644
6645
6646
6647
6648
6649
6650
6651
6652

<1>
<1> mov byte [DISK_STATUS1], TIME_OUT ; ERROR
<1> stc
<1> WQ_OK:
<1> retn
<1> ;WQ_OK:
<1> ;clc
<1> ;ret
<1>
<1> ; 16/07/2022 - TRDOS 386 v2.0.5
<1>
<1> ;-----
<1> ; CHECK FIXED DISK STATUS BYTE :
<1> ;-----
<1> CHECK_ST:
<1> ;mov dx, HF_PORT+7 ; GET THE STATUS
<1> mov dx, [HF_PORT]
<1> add dl, 7
<1>
<1> ; 17/02/2016
<1> ;(http://wiki.osdev.org/ATA_PIO_Mode)
<1> ;"delay 400ns to allow drive to set new values of BSY and DRQ"
<1> in al, dx
<1> ;in al, dx ; 100ns
<1> ;in al, dx ; 100ns
<1> ;in al, dx ; 100ns
<1> NEWIODELAY ; 18/02/2016 (AWARD BIOS - 1999, 'CKST' in AHSK.ASM)
<2> out 0EBh, al
<1> ;
<1>
<1> ; 16/07/2022
<1> mov [HF_STATUS], al
<1> ;mov ah, 0
<1> sub ah, ah ; 0
<1> test al, ST_BUSY ; IF STILL BUSY
<1> jnz short CKST_EXIT ; REPORT OK
<1> mov ah, WRITE_FAULT
<1> test al, ST_WRT_FLT ; CHECK FOR WRITE FAULT
<1> jnz short CKST_EXIT
<1> mov ah, NOT_RDY
<1> test al, ST_READY ; CHECK FOR NOT READY
<1> jz short CKST_EXIT
<1> mov ah, BAD_SEEK
<1> test al, ST_SEEK_COMPL ; CHECK FOR SEEK NOT COMPLETE
<1> jz short CKST_EXIT
<1> mov ah, DATA_CORRECTED
<1> test al, ST_CORRCTD ; CHECK FOR CORRECTED ECC
<1> jnz short CKST_EXIT
<1> ;mov ah, 0
<1> xor ah, ah ; 0
<1> CKST_EXIT:
<1> mov [DISK_STATUS1], ah ; SET ERROR FLAG
<1> cmp ah, DATA_CORRECTED ; KEEP GOING WITH DATA CORRECTED
<1> je short CKST_EX1
<1> ;cmp ah, 0
<1> and ah, ah
<1> CKST_EX1:
<1> retn
<1>
<1> ; 16/07/2022 - TRDOS 386 v2.0.5
<1>
<1> ;-----
<1> ; CHECK FIXED DISK ERROR REGISTER :
<1> ;-----
<1> CHECK_ER:
<1> ;mov dx, HF_PORT+1 ; GET THE ERROR REGISTER
<1> mov dx, [HF_PORT]
<1> inc dl
<1> in al, dx
<1> ; 16/07/2022
<1> ;mov [HF_ERROR], al
<1> ;push ebx ; 21/02/2015
<1> sub ecx, ecx
<1> ;mov ecx, 8 ; TEST ALL 8 BITS
<1> mov cl, 8
<1> CK1:
<1> shl al, 1 ; MOVE NEXT ERROR BIT TO CARRY
<1> jc short CK2 ; FOUND THE ERROR
<1> loop CK1 ; KEEP TRYING
<1> CK2:
<1> ;mov ebx, ERR_TBL ; COMPUTE ADDRESS OF
<1> ;add ebx, ecx ; ERROR CODE
<1> add ecx, ERR_TBL ; 16/07/2022
<1>
<1> ;;;mov ah, byte [cs:bx] ; GET ERROR CODE
<1> ;mov ah, [bx]
<1> ;mov ah, [ebx] ; 21/02/2015
<1> mov ah, [ecx]
<1> CKEX:
<1> mov [DISK_STATUS1], ah ; SAVE ERROR CODE
<1> ; 16/07/2022
<1> ;pop ebx
<1> ;cmp ah, 0
<1> ;and ah, ah
<1> retn
<1>
<1> ;-----
<1> ; CHECK_DMA :
<1> ; -CHECK ES:BX AND # SECTORS TO MAKE SURE THAT IT WILL :
<1> ; FIT WITHOUT SEGMENT OVERFLOW. :
<1> ; -ES:BX HAS BEEN REVISED TO THE FORMAT SSSS:000X :
<1> ; -OK IF # SECTORS < 80H (7FH IF LONG READ OR WRITE) :
<1> ; -OK IF # SECTORS = 80H (7FH) AND BX <= 00H (04H) :
<1> ; -ERROR OTHERWISE :
<1> ;-----
<1>
<1> ; 16/07/2022 - TRDOS 386 v2.0.5
<1> ; (not needed for hard disks and 32 bit os)
<1>
<1> ;CHECK_DMA:
<1> ; 11/04/2021 (32 bit push/pop)
<1> ; push eax ; SAVE REGISTERS
<1> ; mov ax, 8000h ; AH = MAX # SECTORS
<1> ; ; AL = MAX OFFSET
<1> ; test byte [CMD_BLOCK+6], ECC_MODE
<1> ; jz short CKD1
<1> ; mov ah, 7F04h ; ECC IS 4 MORE BYTES
<1> ;CKD1:
<1> ; cmp ah, [CMD_BLOCK+1] ; NUMBER OF SECTORS
<1> ; ja short CKDOK ; IT WILL FIT
<1> ; jb short CKDERR ; TOO MANY
<1> ; cmp al, bl ; CHECK OFFSET ON MAX SECTORS
<1> ; ;jb short CKDERR ; ERROR
<1> ; jnb short CKDOK ; 16/07/2022

```



```

6653 <1> ;CKDOK:
6654 <1> ; ;clc ; CLEAR CARRY
6655 <1> ; ;pop eax
6656 <1> ; ;retn ; NORMAL RETURN
6657 <1> ;CKDERR:
6658 <1> ; ;stc ; INDICATE ERROR
6659 <1> ; ;mov byte [DISK_STATUS1], DMA_BOUNDARY
6660 <1> ;CKDOK:
6661 <1> ; ;pop eax
6662 <1> ; ;retn
6663 <1>
6664 <1>
6665 <1> ;-----
6666 <1> ; SET UP EBX-> DISK PARMS :
6667 <1> ;-----
6668 <1> ; INPUT -> DL = 0 based drive number
6669 <1> ; OUTPUT -> EBX = disk parameter table address
6670 <1>
6671 <1> GET_VEC:
6672 <1> ;sub ax, ax ; GET DISK PARAMETER ADDRESS
6673 <1> ;mov es, ax
6674 <1> ;test dl, 1
6675 <1> ;jz short GV_0
6676 <1> ;les bx, [HF1_TBL_VEC] ; ES:BX -> DRIVE PARAMETERS
6677 <1> ;jmp short GV_EXIT
6678 <1> ;GV_0:
6679 <1> ;les bx, [HF_TBL_VEC] ; ES:BX -> DRIVE PARAMETERS
6680 <1> ;
6681 000057CF 31DB <1> xor ebx, ebx
6682 000057D1 88D3 <1> mov bl, dl
6683 <1> ;02/01/2015
6684 <1> ;xor bh, bh
6685 <1> ;shl bl, 1 ; port address offset
6686 <1> ;mov ax, [bx+hd_ports] ; Base port address (1F0h, 170h)
6687 <1> ;shl bl, 1 ; dpt pointer offset
6688 000057D3 C0E302 <1> shl bl, 2
6689 <1> ;add bx, HF_TBL_VEC ; Disk parameter table pointer
6690 000057D6 81C3[3C770100] <1> add ebx, HF_TBL_VEC ; 21/02/2015
6691 <1> ;push word [bx+2] ; dpt segment
6692 <1> ;pop es
6693 <1> ;mov bx, [bx] ; dpt offset
6694 000057DC 8B1B <1> mov ebx, [ebx]
6695 <1> ;GV_EXIT:
6696 000057DE C3 <1> retn
6697 <1>
6698 <1> hdc1_int: ; 21/02/2015
6699 <1> ;--- HARDWARE INT 76H -- ( IRQ LEVEL 14 ) -----
6700 <1> ;
6701 <1> ; FIXED DISK INTERRUPT ROUTINE :
6702 <1> ; :
6703 <1> ;-----
6704 <1>
6705 <1> ; 22/12/2014
6706 <1> ; IBM PC-XT Model 286 System BIOS Source Code - DISK.ASM (HD_INT)
6707 <1> ; '11/15/85'
6708 <1> ; AWARD BIOS 1999 (D1A0622)
6709 <1> ; Source Code - ATORGS.ASM (INT_HDISK, INT_HDISK1)
6710 <1>
6711 <1> ;int_76h:
6712 <1> HD_INT:
6713 <1> ; 11/04/2021 (32 bit push/pop)
6714 000057DF 50 <1> push eax
6715 000057E0 1E <1> push ds
6716 <1> ;call DDS
6717 <1> ; 21/02/2015 (32 bit, 386 pm modification)
6718 000057E1 66B81000 <1> mov ax, KDATA
6719 000057E5 8ED8 <1> mov ds, ax
6720 <1> ;
6721 <1> ; ;mov @HF_INT_FLAG, 0FFh ; ALL DONE
6722 <1> ; ;mov byte [CS:HF_INT_FLAG], 0FFh
6723 000057E7 C605[32770100]FF <1> mov byte [HF_INT_FLAG], 0FFh
6724 <1> ;
6725 000057EE 52 <1> push edx
6726 000057EF 66BAF701 <1> mov dx, HDC1_BASEPORT+7 ; Status Register (1F7h)
6727 <1> ; Clear Controller
6728 <1> ; (Award BIOS - 1999)
6729 000057F3 EC <1> Clear_IRQ1415:
6730 000057F4 5A <1> in al, dx ;
6731 <1> pop edx
6732 <2> NEWIODELAY
6733 000057F5 E6EB <1> out 0EBh, al
6734 <1> ;
6735 000057F7 B020 <1> mov al, EOI ; NON-SPECIFIC END OF INTERRUPT
6736 000057F9 E6A0 <1> out INTB00, al ; FOR CONTROLLER #2
6737 <1> ;jmp $+2 ; WAIT
6738 <1> NEWIODELAY
6739 000057FB E6EB <2> out 0EBh, al
6740 000057FD E620 <1> out INTA00, al ; FOR CONTROLLER #1
6741 000057FF 1F <1> pop ds
6742 <1> ;sti ; RE-ENABLE INTERRUPTS
6743 <1> ;mov ax, 9100h ; DEVICE POST
6744 <1> ;int 15h ; INTERRUPT
6745 <1> irq15_iret: ; 25/02/2015
6746 00005800 58 <1> pop eax
6747 00005801 CF <1> iretd ; RETURN FROM INTERRUPT
6748 <1>
6749 <1> hdc2_int: ; 21/02/2015
6750 <1> ;--- HARDWARE INT 77H -- ( IRQ LEVEL 15 ) -----
6751 <1> ;
6752 <1> ; FIXED DISK INTERRUPT ROUTINE :
6753 <1> ; :
6754 <1> ;-----
6755 <1> ;int_77h:
6756 <1> HD1_INT:
6757 <1> ; 11/04/2021 (32 bit push/pop)
6758 00005802 50 <1> push eax
6759 <1> ; Check if that is a spurious IRQ (from slave PIC)
6760 00005803 B00B <1> ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
6761 00005805 E6A0 <1> mov al, 0Bh ; In-Service Register
6762 00005807 EB00 <1> out 0A0h, al
6763 00005809 EB00 <1> jmp short $+2
6764 0000580B E4A0 <1> jmp short $+2
6765 0000580D 2480 <1> in al, 0A0h
6766 0000580F 74EF <1> and al, 80h ; bit 7 (is it real IRQ 15 or fake?)
6767 <1> jz short irq15_iret ; Fake (spurious) IRQ, do not send EOI
6768 <1> ;
6769 00005811 1E <1> push ds
6770 <1> ;call DDS
6771 00005812 66B81000 <1> ; 21/02/2015 (32 bit, 386 pm modification)
6772 00005816 8ED8 <1> mov ax, KDATA
6773 <1> mov ds, ax
6774 <1> ;
6775 <1> ; ;mov @HF_INT_FLAG, 0FFh ; ALL DONE
6776 <1> ; ;or byte [CS:HF_INT_FLAG], 0C0h

```

```

6775 00005818 800D[32770100]C0 <1> or byte [HF_INT_FLAG], 0C0h
6776 <1> ;
6777 0000581F 52 <1> push edx
6778 00005820 66BA7701 <1> mov dx, HDC2_BASEPORT+7 ; Status Register (177h)
6779 <1> ; Clear Controller (Award BIOS 1999)
6780 00005824 EB CD <1> jmp short Clear_IRQ1415
6781 <1>
6782 <1> ;%include 'diskdata.inc' ; 11/03/2015
6783 <1> ;%include 'diskbss.inc' ; 11/03/2015
6784 <1>
6785 <1> ;////////////////////
6786 <1> ; END OF DISK I/O SYTEM ///
3228 <1> %include 'memory.s' ; 09/03/2015
<1> ; *****
<1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.8 - memory.s
<1> ; -----
<1> ; Last Update: 04/06/2024 (Previous: 02/12/2023 - Kernel v2.0.7)
<1> ; -----
<1> ; Beginning: 24/01/2016
<1> ; -----
<1> ; Assembler: NASM version 2.15 (trdos386.s)
<1> ; -----
<1> ; Turkish Rational DOS
<1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
<1> ;
<1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
<1> ; memory.inc (18/10/2015)
<1> ; *****
<1> ;
<1> ; MEMORY.ASM - Retro UNIX 386 v1 MEMORY MANAGEMENT FUNCTIONS (PROCEDURES)
<1> ; Retro UNIX 386 v1 Kernel (unix386.s, v0.2.0.14) - MEMORY.INC
<1> ; Last Modification: 18/10/2015
<1> ;
<1> ; ////////////////// MEMORY MANAGEMENT FUNCTIONS (PROCEDURES) //////////////////
<1> ;
<1> ; ;04/11/2014 (unix386.s)
<1> ; PDE_A_PRESENT equ 1 ; Present flag for PDE
<1> ; PDE_A_WRITE equ 2 ; Writable (write permission) flag
<1> ; PDE_A_USER equ 4 ; User (non-system/kernel) page flag
<1> ;
<1> ; PTE_A_PRESENT equ 1 ; Present flag for PTE (bit 0)
<1> ; PTE_A_WRITE equ 2 ; Writable (write permission) flag (bit 1)
<1> ; PTE_A_USER equ 4 ; User (non-system/kernel) page flag (bit 2)
<1> ; PTE_A_ACCESS equ 32 ; Accessed flag (bit 5) ; 09/03/2015
<1> ;
<1> ; 27/04/2015
<1> ; 09/03/2015
<1> PAGE_SIZE equ 4096 ; page size in bytes
<1> PAGE_SHIFT equ 12 ; page table shift count
<1> PAGE_D_SHIFT equ 22 ; 12 + 10 ; page directory shift count
<1> PAGE_OFF equ 0FFFh ; 12 bit byte offset in page frame
<1> PTE_MASK equ 03FFh ; page table entry mask
<1> PTE_DUPLICATED equ 200h ; duplicated page sign (AVL bit 0)
<1> PDE_A_CLEAR equ 0F000h ; to clear PDE attribute bits
<1> PTE_A_CLEAR equ 0F000h ; to clear PTE attribute bits
<1> LOGIC_SECT_SIZE equ 512 ; logical sector size
<1> ERR_MAJOR_PF equ 0E0h ; major error: page fault
<1> ERR_MINOR_IM equ 4 ;15/10/2016 (1->4); insufficient (out of) memory
<1> ERR_MINOR_PV equ 6 ;15/10/2016 (1->4); protection violation
<1> SWP_DISK_READ_ERR equ 40
<1> SWP_DISK_NOT_PRESENT_ERR equ 41
<1> SWP_SECTOR_NOT_PRESENT_ERR equ 42
<1> SWP_NO_FREE_SPACE_ERR equ 43
<1> SWP_DISK_WRITE_ERR equ 44
<1> SWP_NO_PAGE_TO_SWAP_ERR equ 45
<1> PTE_A_ACCESS_BIT equ 5 ; Bit 5 (accessed flag)
<1> SECTOR_SHIFT equ 3 ; sector shift (to convert page block number)
<1> ; 12/07/2016
<1> PTE_SHARED equ 400h ; AVL bit 1, direct memory access bit
<1> ; (Indicates that the page is not allocated
<1> ; for the process, it is a shared or system
<1> ; page, it must not be deallocated!)
<1> ;
<1> ; 14/12/2020
<1> ; (Linear Frame Buffer - video memory mark : AVL bit 1, outside M.A.T.)
<1> PDE_EXTERNAL equ 400h ; Page directory entry for external memory blocks
<1> PTE_EXTERNAL equ 400h ; Allocated kernel pages for Linear Frame Buffer
<1> ; (Out of memory allocation table)
<1> ;
<1> ; Retro Unix 386 v1 - paging method/principles
<1> ;
<1> ; 10/10/2014
<1> ; RETRO UNIX 386 v1 - PAGING METHOD/PRINCIPLES
<1> ;
<1> ; KERNEL PAGE MAP: 1 to 1 physical memory page map
<1> ; (virtual address = physical address)
<1> ; KERNEL PAGE TABLES:
<1> ; Kernel page directory and all page tables are
<1> ; on memory as initialized, as equal to physical memory
<1> ; layout. Kernel pages can/must not be swapped out/in.
<1> ;
<1> ; what for: User pages may be swapped out, when accessing
<1> ; a page in kernel/system mode, if it would be swapped out,
<1> ; kernel would have to swap it in! But it is also may be
<1> ; in use by a user process. (In system/kernel mode
<1> ; kernel can access all memory pages even if they are
<1> ; reserved/allocated for user processes. Swap out/in would
<1> ; cause conflicts.)
<1> ;
<1> ; As result of these conditions,
<1> ; all kernel pages must be initialized as equal to
<1> ; physical layout for preventing page faults.
<1> ; Also, calling "allocate page" procedure after
<1> ; a page fault can cause another page fault (double fault)
<1> ; if all kernel page tables would not be initialized.
<1> ;
<1> ; [first_page] = Beginning of users space, as offset to
<1> ; memory allocation table. (double word aligned)
<1> ;
<1> ; [next_page] = first/next free space to be searched
<1> ; as offset to memory allocation table. (dw aligned)
<1> ;
<1> ; [last_page] = End of memory (users space), as offset
<1> ; to memory allocation table. (double word aligned)
<1> ;
<1> ; USER PAGE TABLES:
<1> ; Demand paging (& 'copy on write' allocation method) ...
<1> ; 'ready only' marked copies of the
<1> ; parent process's page table entries (for
<1> ; same physical memory).
<1> ; (A page will be copied to a new page after
<1> ; if it causes R/W page fault.)
<1> ;
<1> ; Every user process has own (different)

```

```

112 <1> ;; page directory and page tables.
113 <1> ;;
114 <1> ;; Code starts at virtual address 0, always.
115 <1> ;; (Initial value of EIP is 0 in user mode.)
116 <1> ;; (Programs can be written/developed as simple
117 <1> ;; flat memory programs.)
118 <1> ;;
119 <1> ;; MEMORY ALLOCATION STRATEGY:
120 <1> ;; Memory page will be allocated by kernel only
121 <1> ;; (in kernel/system mode only).
122 <1> ;; * After a
123 <1> ;; - 'not present' page fault
124 <1> ;; - 'writing attempt on read only page' page fault
125 <1> ;; * For loading (opening, reading) a file or disk/drive
126 <1> ;; * As response to 'allocate additional memory blocks'
127 <1> ;; request by running process.
128 <1> ;; * While creating a process, allocating a new buffer,
129 <1> ;; new page tables etc.
130 <1> ;;
131 <1> ;; At first,
132 <1> ;; - 'allocate page' procedure will be called;
133 <1> ;; if it will return with a valid (>0) physical address
134 <1> ;; (that means the relevant M.A.T. bit has been RESET)
135 <1> ;; relevant memory page/block will be cleared (zeroed).
136 <1> ;; - 'allocate page' will be called for allocating page
137 <1> ;; directory, page table and running space (data/code).
138 <1> ;; - every successful 'allocate page' call will decrease
139 <1> ;; 'free_pages' count (pointer).
140 <1> ;; - 'out of (insufficient) memory error' will be returned
141 <1> ;; if 'free_pages' points to a ZERO.
142 <1> ;; - swapping out and swapping in (if it is not a new page)
143 <1> ;; procedures will be called as response to 'out of memory'
144 <1> ;; error except errors caused by attribute conflicts.
145 <1> ;; (swapper functions)
146 <1> ;;
147 <1> ;; At second,
148 <1> ;; - page directory entry will be updated then page table
149 <1> ;; entry will be updated.
150 <1> ;;
151 <1> ;; MEMORY ALLOCATION TABLE FORMAT:
152 <1> ;; - M.A.T. has a size according to available memory as
153 <1> ;; follows:
154 <1> ;; - 1 (allocation) bit per 1 page (4096 bytes)
155 <1> ;; - a bit with value of 0 means allocated page
156 <1> ;; - a bit with value of 1 means a free page
157 <1> ;; - 'free_pages' pointer holds count of free pages
158 <1> ;; depending on M.A.T.
159 <1> ;; (NOTE: Free page count will not be checked
160 <1> ;; again -on M.A.T.- after initialization.
161 <1> ;; Kernel will trust on initial count.)
162 <1> ;; - 'free_pages' count will be decreased by allocation
163 <1> ;; and it will be increased by deallocation procedures.
164 <1> ;;
165 <1> ;; - Available memory will be calculated during
166 <1> ;; the kernel's initialization stage (in real mode).
167 <1> ;; Memory allocation table and kernel page tables
168 <1> ;; will be formatted/sized as result of available
169 <1> ;; memory calculation before paging is enabled.
170 <1> ;;
171 <1> ;; For 4GB Available/Present Memory: (max. possible memory size)
172 <1> ;; - Memory Allocation Table size will be 128 KB.
173 <1> ;; - Memory allocation for kernel page directory size
174 <1> ;; is always 4 KB. (in addition to total allocation size
175 <1> ;; for page tables)
176 <1> ;; - Memory allocation for kernel page tables (1024 tables)
177 <1> ;; is 4 MB (1024*4*1024 bytes).
178 <1> ;; - User (available) space will be started
179 <1> ;; at 6th MB of the memory (after 1MB+4MB).
180 <1> ;; - The first 640 KB is for kernel's itself plus
181 <1> ;; memory allocation table and kernel's page directory
182 <1> ;; (D0000h-EFFFFh may be used as kernel space...)
183 <1> ;; - B0000h to B7FFFh address space (32 KB) will be used
184 <1> ;; for buffers.
185 <1> ;; - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
186 <1> ;; (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
187 <1> ;; - Kernel page tables start at 100000h (2nd MB)
188 <1> ;;
189 <1> ;; For 1GB Available Memory:
190 <1> ;; - Memory Allocation Table size will be 32 KB.
191 <1> ;; - Memory allocation for kernel page directory size
192 <1> ;; is always 4 KB. (in addition to total allocation size
193 <1> ;; for page tables)
194 <1> ;; - Memory allocation for kernel page tables (256 tables)
195 <1> ;; is 1 MB (256*4*1024 bytes).
196 <1> ;; - User (available) space will be started
197 <1> ;; at 3th MB of the memory (after 1MB+1MB).
198 <1> ;; - The first 640 KB is for kernel's itself plus
199 <1> ;; memory allocation table and kernel's page directory
200 <1> ;; (D0000h-EFFFFh may be used as kernel space...)
201 <1> ;; - B0000h to B7FFFh address space (32 KB) will be used
202 <1> ;; for buffers.
203 <1> ;; - ROMBIOS, VIDEO BUFFER and VIDEO ROM space are reserved.
204 <1> ;; (A0000h-AFFFFh, C0000h-CFFFFh, F0000h-FFFFFh)
205 <1> ;; - Kernel page tables start at 100000h (2nd MB).
206 <1> ;;
207 <1> ;;
208 <1> ;;
209 <1> ;;
210 <1> ;; *****
211 <1> ;;
212 <1> ;; RETRO UNIX 386 v1 - Paging (Method for Copy On Write paging principle)
213 <1> ;; DEMAND PAGING - PARENT&CHILD PAGE TABLE DUPLICATION PRINCIPLES (23/04/2015)
214 <1> ;;
215 <1> ;; Main factor: "sys fork" system call
216 <1> ;;
217 <1> ;; FORK
218 <1> ;; |-----> parent - duplicated PTEs, read only pages
219 <1> ;; writable pages ----> |
220 <1> ;; |-----> child - duplicated PTEs, read only pages
221 <1> ;;
222 <1> ;; AVL bit (0) of Page Table Entry is used as duplication sign
223 <1> ;;
224 <1> ;; AVL Bit 0 [PTE Bit 9] = 'Duplicated PTE belongs to child' sign/flag (if it is set)
225 <1> ;; Note: Dirty bit (PTE bit 6) may be used instead of AVL bit 0 (PTE bit 9)
226 <1> ;; -while R/W bit is 0-.
227 <1> ;;
228 <1> ;; Duplicate page tables with writable pages (the 1st sys fork in the process):
229 <1> ;; # Parent's Page Table Entries are updated to point same pages as read only,
230 <1> ;; as duplicated PTE bit -AVL bit 0, PTE bit 9- are reset/clear.
231 <1> ;; # Then Parent's Page Table is copied to Child's Page Table.
232 <1> ;; # Child's Page Table Entries are updated as duplicated child bit
233 <1> ;; -AVL bit 0, PTE bit 9- is set.
234 <1> ;;
235 <1> ;; Duplicate page tables with read only pages (several sys fork system calls):

```

```

236 <1> ; # Parent's read only pages are copied to new child pages.
237 <1> ; Parent's PTE attributes are not changed.
238 <1> ; (Because, there is another parent-child fork before this fork! We must not
239 <1> ; destroy/mix previous fork result).
240 <1> ; # Child's Page Table Entries (which are corresponding to Parent's
241 <1> ; read only pages) are set as writable (while duplicated PTE bit is clear).
242 <1> ; # Parent's PTEs with writable page attribute are updated to point same pages
243 <1> ; as read only, (while) duplicated PTE bit is reset (clear).
244 <1> ; # Parent's Page Table Entries (with writable page attribute) are duplicated
245 <1> ; as Child's Page Table Entries without copying actual page.
246 <1> ; # Child's Page Table Entries (which are corresponding to Parent's writable
247 <1> ; pages) are updated as duplicated PTE bit (AVL bit 0, PTE bit 9- is set.
248 <1> ;
249 <1> ; !? WHAT FOR (duplication after duplication):
250 <1> ; In UNIX method for sys fork (a typical 'fork' application in /etc/init)
251 <1> ; program/executable code continues from specified location as child process,
252 <1> ; returns back previous code location as parent process, every child after
253 <1> ; every sys fork uses last image of code and data just prior the fork.
254 <1> ; Even if the parent code changes data, the child will not see the changed data
255 <1> ; after the fork. In Retro UNIX 8086 v1, parent's process segment (32KB)
256 <1> ; was copied to child's process segment (all of code and data) according to
257 <1> ; original UNIX v1 which copies all of parent process code and data -core-
258 <1> ; to child space -core- but swaps that core image -of child- on to disk.
259 <1> ; If I (Erdogan Tan) would use a method of to copy parent's core
260 <1> ; (complete running image of parent process) to the child process;
261 <1> ; for big sizes, i would force Retro UNIX 386 v1 to spend many memory pages
262 <1> ; and times only for a sys fork. (It would excessive reservation for sys fork,
263 <1> ; because sys fork usually is prior to sys exec; sys exec always establishes
264 <1> ; a new/fresh core -running space-, by clearing all code/data content).
265 <1> ; 'Read Only' page flag ensures page fault handler is needed only for a few write
266 <1> ; attempts between sys fork and sys exec, not more... (I say so by thinking
267 <1> ; of "/etc/init" content, specially.) sys exec will clear page tables and
268 <1> ; new/fresh pages will be used to load and run new executable/program.
269 <1> ; That is what for i have preferred "copy on write", "duplication" method
270 <1> ; for sharing same read only pages between parent and child processes.
271 <1> ; That is a pity i have to use new private flag (AVL bit 0, "duplicated PTE
272 <1> ; belongs to child" sign) for cooperation on duplicated pages between a parent
273 <1> ; and it's child processes; otherwise parent process would destroy data belongs
274 <1> ; to its child or vice versa; or some pages would remain unclaimed
275 <1> ; -deallocation problem-.
276 <1> ; Note: to prevent conflicts, read only pages must not be swapped out...
277 <1> ;
278 <1> ; WHEN PARENT TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
279 <1> ; # Page fault handler will do those:
280 <1> ; - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
281 <1> ; - If it is reset/clear, there is a child uses same page.
282 <1> ; - Parent's read only page -previous page- is copied to a new writable page.
283 <1> ; - Parent's PTE is updated as writable page, as unique page (AVL=0)
284 <1> ; - (Page fault handler will check this PTE later, if child process causes to
285 <1> ; page fault due to write attempt on read only page. Of course, the previous
286 <1> ; read only page will be converted to writable and unique page which belongs
287 <1> ; to child process.)
288 <1> ; WHEN CHILD TRIES TO WRITE IT'S READ ONLY (DUPLICATED) PAGE:
289 <1> ; # Page fault handler will do those:
290 <1> ; - 'Duplicated PTE' flag (PTE bit 9) is checked (on the failed PTE).
291 <1> ; - If it is set, there is a parent uses -or was using- same page.
292 <1> ; - Same PTE address within parent's page table is checked if it has same page
293 <1> ; address or not.
294 <1> ; - If parent's PTE has same address, child will continue with a new writable page.
295 <1> ; Parent's PTE will point to same (previous) page as writable, unique (AVL=0).
296 <1> ; - If parent's PTE has different address, child will continue with it's
297 <1> ; own/same page but read only flag (0) will be changed to writable flag (1) and
298 <1> ; 'duplicated PTE (belongs to child)' flag/sign will be cleared/reset.
299 <1> ;
300 <1> ; NOTE: When a child process is terminated, read only flags of parent's page tables
301 <1> ; will be set as writable (and unique) in case of child process was using
302 <1> ; same pages with duplicated child PTE sign... Depending on sys fork and
303 <1> ; duplication method details, it is not possible multiple child processes
304 <1> ; were using same page with duplicated PTEs.
305 <1> ;
306 <1> ; *****
307 <1> ;
308 <1> ; 08/10/2014
309 <1> ; 11/09/2014 - Retro UNIX 386 v1 PAGING (further) draft
310 <1> ; by Erdogan Tan (Based on KolibriOS 'memory.inc')
311 <1> ;
312 <1> ; 'allocate_page' code is derived and modified from KolibriOS
313 <1> ; 'alloc_page' procedure in 'memory.inc'
314 <1> ; (25/08/2014, Revision: 5057) file
315 <1> ; by KolibriOS Team (2004-2012)
316 <1> ;
317 <1> ; allocate_page:
318 <1> ; ; 17/04/2021 - TRDOS 386 v2.0.4
319 <1> ; ; (temporary modifications)
320 <1> ; ; 01/07/2015
321 <1> ; ; 05/05/2015
322 <1> ; ; 30/04/2015
323 <1> ; ; 16/10/2014
324 <1> ; ; 08/10/2014
325 <1> ; ; 09/09/2014 (Retro UNIX 386 v1 - beginning)
326 <1> ; ;
327 <1> ; ; INPUT -> none
328 <1> ; ;
329 <1> ; ; OUTPUT ->
330 <1> ; ; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
331 <1> ; ; (corresponding MEMORY ALLOCATION TABLE bit is RESET)
332 <1> ; ;
333 <1> ; ; CF = 1 and EAX = 0
334 <1> ; ; if there is not a free page to be allocated
335 <1> ; ;
336 <1> ; ; Modified Registers -> none (except EAX)
337 <1> ; ;
338 00005826 A1[A8760100] <1> mov eax, [free_pages]
339 0000582B 21C0 <1> and eax, eax
340 0000582D 7438 <1> jz short out_of_memory
341 <1> ;
342 0000582F 53 <1> push ebx
343 00005830 51 <1> push ecx
344 <1> ;
345 00005831 8B00001000 <1> mov ebx, MEM_ALLOC_TBL ; Memory Allocation Table offset
346 00005836 89D9 <1> mov ecx, ebx
347 <1> ;
348 <1> ; ; NOTE: 32 (first_page) is initial
349 <1> ; ; value of [next_page].
350 <1> ; ; It points to the first available
351 <1> ; ; page block for users (ring 3) ...
352 <1> ; ; (MAT offset 32 = 1024/32)
353 <1> ; ; (at the of the first 4 MB)
353 00005838 031D[AC760100] <1> add ebx, [next_page] ; Free page searching starts from here
354 <1> ; ; next_free_page >> 5
355 0000583E 030D[B0760100] <1> add ecx, [last_page] ; Free page searching ends here
356 <1> ; ; (total_pages - 1) >> 5
357 <1> ;
357 al_p_scan: <1>
358 00005844 39CB <1> cmp ebx, ecx
359 00005846 770A <1> ja short al_p_notfound

```

```

360 <1> ;
361 <1> ; 01/07/2015
362 <1> ; AMD64 Architecture Programmer's Manual
363 <1> ; Volume 3:
364 <1> ; General-Purpose and System Instructions
365 <1> ;
366 <1> ; BSF - Bit Scan Forward
367 <1> ;
368 <1> ; Searches the value in a register or a memory location
369 <1> ; (second operand) for the least-significant set bit.
370 <1> ; If a set bit is found, the instruction clears the zero flag (ZF)
371 <1> ; and stores the index of the least-significant set bit in a destination
372 <1> ; register (first operand). If the second operand contains 0,
373 <1> ; the instruction sets ZF to 1 and does not change the contents of the
374 <1> ; destination register. The bit index is an unsigned offset from bit 0
375 <1> ; of the searched value
376 <1> ;
377 00005848 0FBC03 <1> bsf     eax, [ebx] ; Scans source operand for first bit set (1).
378 <1> ; Clear ZF if a bit is found set (1) and
379 <1> ; loads the destination with an index to
380 <1> ; first set bit. (0 -> 31)
381 <1> ; Sets ZF to 1 if no bits are found set.
382 0000584B 751C <1> jnz     short al_p_found ; ZF = 0 -> a free page has been found
383 <1> ;
384 <1> ; NOTE: a Memory Allocation Table bit
385 <1> ; with value of 1 means
386 <1> ; the corresponding page is free
387 <1> ; (Retro UNIX 386 v1 feature only!)
388 0000584D 83C304 <1> add     ebx, 4
389 <1> ;
390 <1> ; We return back for searching next page block
391 <1> ; NOTE: [free_pages] is not ZERO; so,
392 00005850 EBF2 <1> jmp     short al_p_scan ; we always will find at least 1 free page here.
393 <1> ;
394 <1> al_p_notfound:
395 00005852 81E900001000 <1> sub     ecx, MEM_ALLOC_TBL
396 00005858 890D[AC760100] <1> mov     [next_page], ecx ; next/first free page = last page
397 <1> ; (deallocate_page procedure will change it)
398 0000585E 31C0 <1> xor     eax, eax
399 00005860 A3[A8760100] <1> mov     [free_pages], eax ; 0
400 00005865 59 <1> pop     ecx
401 00005866 5B <1> pop     ebx
402 <1> ;
403 <1> ; 17/04/2021
404 <1> ; ('swap_out' procedure call is disabled as temporary)
405 <1> ;
406 <1> out_of_memory:
407 <1> ; call swap_out
408 <1> ; jnc short al_p_ok ; [free_pages] = 0, re-allocation by swap_out
409 <1> ;
410 <1> ; sub     eax, eax ; 0
411 00005867 F9 <1> stc
412 00005868 C3 <1> retn
413 <1> ;
414 <1> al_p_found:
415 00005869 89D9 <1> mov     ecx, ebx
416 0000586B 81E900001000 <1> sub     ecx, MEM_ALLOC_TBL
417 00005871 890D[AC760100] <1> mov     [next_page], ecx ; Set first free page searching start
418 <1> ; address/offset (to the next)
419 00005877 FF0D[A8760100] <1> dec     dword [free_pages] ; 1 page has been allocated (X = X-1)
420 <1> ;
421 0000587D 0FB303 <1> btr     [ebx], eax ; The destination bit indexed by the source value
422 <1> ; is copied into the Carry Flag and then cleared
423 <1> ; in the destination.
424 <1> ;
425 <1> ; Reset the bit which is corresponding to the
426 <1> ; (just) allocated page.
427 <1> ; 01/07/2015 (4*8 = 32, 1 allocation byte = 8 pages)
428 00005880 C1E103 <1> shl     ecx, 3 ; (page block offset * 32) + page index
429 00005883 01C8 <1> add     eax, ecx ; = page number
430 00005885 C1E00C <1> shl     eax, 12 ; physical address of the page (flat/real value)
431 <1> ; EAX = physical address of memory page
432 <1> ;
433 <1> ; NOTE: The relevant page directory and page table entry will be updated
434 <1> ; according to this EAX value...
435 00005888 59 <1> pop     ecx
436 00005889 5B <1> pop     ebx
437 <1> al_p_ok:
438 0000588A C3 <1> retn
439 <1> ;
440 <1> make_page_dir:
441 <1> ; 18/04/2015
442 <1> ; 12/04/2015
443 <1> ; 23/10/2014
444 <1> ; 16/10/2014
445 <1> ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
446 <1> ;
447 <1> ; INPUT ->
448 <1> ; none
449 <1> ; OUTPUT ->
450 <1> ; (EAX = 0)
451 <1> ; cf = 1 -> insufficient (out of) memory error
452 <1> ; cf = 0 ->
453 <1> ; u.pgdir = page directory (physical) address of the current
454 <1> ; process/user.
455 <1> ;
456 <1> ; Modified Registers -> EAX
457 <1> ;
458 0000588B E896FFFFFF <1> call    allocate_page
459 00005890 7216 <1> jc      short mkpd_error
460 <1> ;
461 00005892 A3[AC8E0100] <1> mov     [u.pgdir], eax ; Page dir address for current user/process
462 <1> ; (Physical address)
463 <1> clear_page:
464 <1> ; 18/04/2015
465 <1> ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
466 <1> ;
467 <1> ; INPUT ->
468 <1> ; EAX = physical address of the page
469 <1> ; OUTPUT ->
470 <1> ; all bytes of the page will be cleared
471 <1> ;
472 <1> ; Modified Registers -> none
473 <1> ;
474 00005897 57 <1> push    edi
475 00005898 51 <1> push    ecx
476 00005899 50 <1> push    eax
477 0000589A B900040000 <1> mov     ecx, PAGE_SIZE / 4
478 0000589F 89C7 <1> mov     edi, eax
479 000058A1 31C0 <1> xor     eax, eax
480 000058A3 F3AB <1> rep     stosd
481 000058A5 58 <1> pop     eax
482 000058A6 59 <1> pop     ecx
483 000058A7 5F <1> pop     edi

```

```

484      <1> mkpd_error:
485      <1> mkpt_error:
486 000058A8 C3      <1>      retn
487      <1>
488      <1> make_page_table:
489      <1>      ; 23/06/2015
490      <1>      ; 18/04/2015
491      <1>      ; 12/04/2015
492      <1>      ; 16/10/2014
493      <1>      ; 09/10/2014 ; (Retro UNIX 386 v1 - beginning)
494      <1>
495      <1>      ; INPUT ->
496      <1>      ;     EBX = virtual (linear) address
497      <1>      ;     ECX = page table attributes (lower 12 bits)
498      <1>      ;           (higher 20 bits must be ZERO)
499      <1>      ;           (bit 0 must be 1)
500      <1>      ;     u.pgdir = page directory (physical) address
501      <1>      ; OUTPUT ->
502      <1>      ;     EDX = Page directory entry address
503      <1>      ;     EAX = Page table address
504      <1>      ;     cf = 1 -> insufficient (out of) memory error
505      <1>      ;     cf = 0 -> page table address in the PDE (EDX)
506      <1>
507      <1>      ; Modified Registers -> EAX, EDX
508      <1>
509 000058A9 E878FFFFFF      <1>      call    allocate_page
510 000058AE 72F8          <1>      jc      short mkpt_error
511 000058B0 E811000000      <1>      call    set_pde
512 000058B5 EBE0          <1>      jmp     short clear_page
513      <1>
514      <1> make_page:
515      <1>      ; 24/07/2015
516      <1>      ; 23/06/2015 ; (Retro UNIX 386 v1 - beginning)
517      <1>
518      <1>      ; INPUT ->
519      <1>      ;     EBX = virtual (linear) address
520      <1>      ;     ECX = page attributes (lower 12 bits)
521      <1>      ;           (higher 20 bits must be ZERO)
522      <1>      ;           (bit 0 must be 1)
523      <1>      ;     u.pgdir = page directory (physical) address
524      <1>      ; OUTPUT ->
525      <1>      ;     EBX = Virtual address
526      <1>      ;     (EDX = PTE value)
527      <1>      ;     EAX = Physical address
528      <1>      ;     cf = 1 -> insufficient (out of) memory error
529      <1>
530      <1>      ; Modified Registers -> EAX, EDX
531      <1>
532 000058B7 E86AFFFFFF      <1>      call    allocate_page
533 000058BC 7207          <1>      jc      short mkp_err
534 000058BE E821000000      <1>      call    set_pte
535 000058C3 73D2          <1>      jnc     short clear_page ; 18/04/2015
536      <1> mkp_err:
537 000058C5 C3          <1>      retn
538      <1>
539      <1>
540      <1> set_pde:      ; Set page directory entry (PDE)
541      <1>      ; 20/07/2015
542      <1>      ; 18/04/2015
543      <1>      ; 12/04/2015
544      <1>      ; 23/10/2014
545      <1>      ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
546      <1>
547      <1>      ; INPUT ->
548      <1>      ;     EAX = physical address
549      <1>      ;           (use present value if EAX = 0)
550      <1>      ;     EBX = virtual (linear) address
551      <1>      ;     ECX = page table attributes (lower 12 bits)
552      <1>      ;           (higher 20 bits must be ZERO)
553      <1>      ;           (bit 0 must be 1)
554      <1>      ;     u.pgdir = page directory (physical) address
555      <1>      ; OUTPUT ->
556      <1>      ;     EDX = PDE address
557      <1>      ;     EAX = page table address (physical)
558      <1>      ;     ;(CF=1 -> Invalid page address)
559      <1>
560      <1>      ; Modified Registers -> EDX
561      <1>
562 000058C6 89DA          <1>      mov     edx, ebx
563 000058C8 C1EA16        <1>      shr     edx, PAGE_D_SHIFT ; 22
564 000058CB C1E202        <1>      shl     edx, 2 ; offset to page directory (1024*4)
565 000058CE 0315[AC8E0100] <1>      add     edx, [u.pgdir]
566      <1>
567      <1>      ; and     eax, eax
568 000058D4 21C0          <1>      jnz     short spde_1
569      <1>
570 000058D8 8B02          <1>      mov     eax, [edx] ; old PDE value
571      <1>      ; test    al, 1
572      <1>      ; jz      short spde_2
573 000058DA 662500F0      <1>      and     ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
574      <1> spde_1:
575      <1>      ; and     cx, 0FFFh
576 000058DE 8902          <1>      mov     [edx], eax
577 000058E0 66090A        <1>      or      [edx], cx
578 000058E3 C3          <1>      retn
579      <1> ;spde_2: ; error
580      <1>      ; stc
581      <1>      ; retn
582      <1>
583      <1> set_pte:      ; Set page table entry (PTE)
584      <1>      ; 24/07/2015
585      <1>      ; 20/07/2015
586      <1>      ; 23/06/2015
587      <1>      ; 18/04/2015
588      <1>      ; 12/04/2015
589      <1>      ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
590      <1>
591      <1>      ; INPUT ->
592      <1>      ;     EAX = physical page address
593      <1>      ;           (use present value if EAX = 0)
594      <1>      ;     EBX = virtual (linear) address
595      <1>      ;     ECX = page attributes (lower 12 bits)
596      <1>      ;           (higher 20 bits must be ZERO)
597      <1>      ;           (bit 0 must be 1)
598      <1>      ;     u.pgdir = page directory (physical) address
599      <1>      ; OUTPUT ->
600      <1>      ;     EAX = physical page address
601      <1>      ;     (EDX = PTE value)
602      <1>      ;     EBX = virtual address
603      <1>
604      <1>      ; CF = 1 -> error
605      <1>
606      <1>      ; Modified Registers -> EAX, EDX
607      <1>

```

```

608 000058E4 50          <1>    push    eax
609 000058E5 A1[AC8E0100] <1>    mov     eax, [u.pgdir] ; 20/07/2015
610 000058EA E837000000 <1>    call    get_pde
611          <1>    ; EDX = PDE address
612          <1>    ; EAX = PDE value
613 000058EF 5A          <1>    pop     edx ; physical page address
614 000058F0 722A        <1>    jc      short spte_err ; PDE not present
615          <1>    ;
616 000058F2 53          <1>    push    ebx ; 24/07/2015
617 000058F3 662500F0 <1>    and     ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
618          <1>    ; EDX = PT address (physical)
619 000058F7 C1EB0C      <1>    shr     ebx, PAGE_SHIFT ; 12
620 000058FA 81E3FF030000 <1>    and     ebx, PTE_MASK ; 03FFh
621          <1>    ; clear higher 10 bits (PD bits)
622 00005900 C1E302      <1>    shl     ebx, 2 ; offset to page table (1024*4)
623 00005903 01C3        <1>    add     ebx, eax
624          <1>    ;
625 00005905 8803        <1>    mov     eax, [ebx] ; Old PTE value
626 00005907 A801        <1>    test    al, 1
627 00005909 740C        <1>    jz      short spte_0
628 0000590B 09D2        <1>    or      edx, edx
629 0000590D 750F        <1>    jnz     short spte_1
630 0000590F 662500F0 <1>    and     ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 bits
631 00005913 89C2        <1>    mov     edx, eax
632 00005915 EB09        <1>    jmp     short spte_2
633          <1>    spte_0:
634          <1>    ; If this PTE contains a swap (disk) address,
635          <1>    ; it can be updated by using 'swap_in' procedure
636          <1>    ; only!
637 00005917 21C0        <1>    and     eax, eax
638 00005919 7403        <1>    jz      short spte_1
639          <1>    ; 24/07/2015
640          <1>    ; swapped page ! (on disk)
641 0000591B 5B          <1>    pop     ebx
642          <1>    spte_err:
643          <1>    stc
644 0000591D C3          <1>    retn
645          <1>    spte_1:
646 0000591E 89D0        <1>    mov     eax, edx
647          <1>    spte_2:
648          <1>    or      edx, ecx
649          <1>    ; 23/06/2015
650 00005922 8913        <1>    mov     [ebx], edx ; PTE value in EDX
651          <1>    ; 24/07/2015
652 00005924 5B          <1>    pop     ebx
653 00005925 C3          <1>    retn
654          <1>
655          <1>    get_pde: ; Get present value of the relevant PDE
656          <1>    ; 20/07/2015
657          <1>    ; 18/04/2015
658          <1>    ; 12/04/2015
659          <1>    ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
660          <1>    ;
661          <1>    ; INPUT ->
662          <1>    ; EBX = virtual (linear) address
663          <1>    ; EAX = page directory (physical) address
664          <1>    ; OUTPUT ->
665          <1>    ; EDX = Page directory entry address
666          <1>    ; EAX = Page directory entry value
667          <1>    ; CF = 1 -> PDE not present or invalid ?
668          <1>    ; Modified Registers -> EDX, EAX
669          <1>    ;
670 00005926 89DA        <1>    mov     edx, ebx
671 00005928 C1EA16      <1>    shr     edx, PAGE_D_SHIFT ; 22 (12+10)
672 0000592B C1E202      <1>    shl     edx, 2 ; offset to page directory (1024*4)
673 0000592E 01C2        <1>    add     edx, eax ; page directory address (physical)
674 00005930 8B02        <1>    mov     eax, [edx]
675 00005932 A801        <1>    test    al, PDE_A_PRESENT ; page table is present or not !
676 00005934 751F        <1>    jnz     short gpde_retn
677 00005936 F9          <1>    stc
678          <1>    gpde_retn:
679 00005937 C3          <1>    retn
680          <1>
681          <1>    get_pte: ; Get present value of the relevant PTE
682          <1>    ; 29/07/2015
683          <1>    ; 20/07/2015
684          <1>    ; 18/04/2015
685          <1>    ; 12/04/2015
686          <1>    ; 10/10/2014 ; (Retro UNIX 386 v1 - beginning)
687          <1>    ;
688          <1>    ; INPUT ->
689          <1>    ; EBX = virtual (linear) address
690          <1>    ; EAX = page directory (physical) address
691          <1>    ; OUTPUT ->
692          <1>    ; EDX = Page table entry address (if CF=0)
693          <1>    ; Page directory entry address (if CF=1)
694          <1>    ; (Bit 0 value is 0 if PT is not present)
695          <1>    ; EAX = Page table entry value (page address)
696          <1>    ; CF = 1 -> PDE not present or invalid ?
697          <1>    ; Modified Registers -> EAX, EDX
698          <1>    ;
699          <1>    ;
700 00005938 E8E9FFFFFF <1>    call    get_pde
701 0000593D 72F8        <1>    jc      short gpde_retn ; page table is not present
702          <1>    ;jnc    short gpte_1
703          <1>    ;retn
704          <1>    ;gpte_1:
705 0000593F 662500F0 <1>    and     ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 bits
706 00005943 89DA        <1>    mov     edx, ebx
707 00005945 C1EA0C      <1>    shr     edx, PAGE_SHIFT ; 12
708 00005948 81E2FF030000 <1>    and     edx, PTE_MASK ; 03FFh
709          <1>    ; clear higher 10 bits (PD bits)
710 0000594E C1E202      <1>    shl     edx, 2 ; offset from start of page table (1024*4)
711 00005951 01C2        <1>    add     edx, eax
712 00005953 8B02        <1>    mov     eax, [edx]
713          <1>    gpte_retn:
714 00005955 C3          <1>    retn
715          <1>
716          <1>    deallocate_page_dir:
717          <1>    ; 15/09/2015
718          <1>    ; 05/08/2015
719          <1>    ; 30/04/2015
720          <1>    ; 28/04/2015
721          <1>    ; 17/10/2014
722          <1>    ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
723          <1>    ;
724          <1>    ; INPUT ->
725          <1>    ; EAX = PHYSICAL ADDRESS OF THE PAGE DIRECTORY (CHILD)
726          <1>    ; EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
727          <1>    ; OUTPUT ->
728          <1>    ; All of page tables in the page directory
729          <1>    ; and page dir's itself will be deallocated
730          <1>    ; except 'read only' duplicated pages (will be converted
731          <1>    ; to writable pages).

```

```

732      <1>      ;
733      <1>      ; Modified Registers -> EAX
734      <1>      ;
735      <1>      ;
736      00005956 56      <1>      push     esi
737      00005957 51      <1>      push     ecx
738      00005958 50      <1>      push     eax
739      00005959 89C6    <1>      mov      esi, eax
740      0000595B 31C9    <1>      xor      ecx, ecx
741      <1>      ; The 1st PDE points to Kernel Page Table 0 (the 1st 4MB),
742      <1>      ; it must not be deallocated
743      0000595D 890E    <1>      mov      [esi], ecx ; 0 ; clear PDE 0
744      <1>      dapt_0:
745      0000595F AD      <1>      lodsd
746      00005960 A801    <1>      test     al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
747      00005962 7409    <1>      jz       short dapt_1
748      00005964 662500F0 <1>      and      ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
749      00005968 E812000000 <1>      call     deallocate_page_table
750      <1>      dapt_1:
751      0000596D 41      <1>      inc      ecx ; page directory entry index
752      0000596E 81F900040000 <1>      cmp      ecx, PAGE_SIZE / 4 ; 1024
753      00005974 72E9    <1>      jb       short dapt_0
754      <1>      dapt_2:
755      00005976 58      <1>      pop      eax
756      00005977 E869000000 <1>      call     deallocate_page ; deallocate the page dir's itself
757      0000597C 59      <1>      pop      ecx
758      0000597D 5E      <1>      pop      esi
759      0000597E C3      <1>      retn
760      <1>
761      <1>      deallocate_page_table:
762      <1>      ; 29/08/2023 - TRDOS 386 v2.0.6
763      <1>      ; 17/04/2021 - TRDOS 386 v2.0.4
764      <1>      ; (temporary modifications)
765      <1>      ; 12/07/2016
766      <1>      ; 19/09/2015
767      <1>      ; 15/09/2015
768      <1>      ; 05/08/2015
769      <1>      ; 30/04/2015
770      <1>      ; 28/04/2015
771      <1>      ; 24/10/2014
772      <1>      ; 23/10/2014
773      <1>      ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
774      <1>
775      <1>      ; INPUT ->
776      <1>      ; EAX = PHYSICAL (real/flat) ADDRESS OF THE PAGE TABLE
777      <1>      ; EBX = PHYSICAL ADDRESS OF THE PARENT'S PAGE DIRECTORY
778      <1>      ; (ECX = page directory entry index)
779      <1>      ; OUTPUT ->
780      <1>      ; All of pages in the page table and page table's itself
781      <1>      ; will be deallocated except 'read only' duplicated pages
782      <1>      ; (will be converted to writable pages).
783      <1>
784      <1>      ; Modified Registers -> EAX
785      <1>      ;
786      0000597F 56      <1>      push     esi
787      00005980 57      <1>      push     edi
788      00005981 52      <1>      push     edx
789      00005982 50      <1>      push     eax ; *
790      00005983 89C6    <1>      mov      esi, eax
791      00005985 31FF    <1>      xor      edi, edi ; 0
792      <1>      dapt_0:
793      00005987 AD      <1>      lodsd
794      00005988 A801    <1>      test     al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
795      0000598A 744C    <1>      jz       short dapt_1
796      <1>      ;
797      0000598C A802    <1>      test     al, PTE_A_WRITE ; bit 1, writable (r/w) flag
798      <1>      ; (must be 1)
799      0000598E 753D    <1>      jnz      short dapt_3
800      <1>      ; Read only -duplicated- page (belongs to a parent or a child)
801      00005990 66A90002 <1>      test     ax, PTE_DUPLICATED ; was this page duplicated
802      <1>      ; as child's page ?
803      00005994 7442    <1>      jz       short dapt_4 ; clear PTE but don't deallocate the page!
804      <1>      ; check the parent's PTE value is read only & same page or not..
805      <1>      ; ECX = page directory entry index (0-1023)
806      00005996 53      <1>      push     ebx
807      00005997 51      <1>      push     ecx
808      <1>      ; shl     cx, 2 ; *4
809      <1>      ; 29/08/2023
810      00005998 C1E102 <1>      shl     ecx, 2
811      0000599B 01CB    <1>      add      ebx, ecx ; PDE offset (for the parent)
812      0000599D 8B0B    <1>      mov      ecx, [ebx]
813      0000599F F6C101 <1>      test     cl, PDE_A_PRESENT ; present (valid) or not ?
814      000059A2 7427    <1>      jz       short dapt_2 ; parent process does not use this page
815      000059A4 6681E100F0 <1>      and      cx, PDE_A_CLEAR ; 0F000h ; clear attribute bits
816      <1>      ; EDI = page table entry index (0-1023)
817      000059A9 89FA    <1>      mov      edx, edi
818      <1>      ; shl     dx, 2 ; *4
819      <1>      ; 29/08/2023
820      000059AB C1E202 <1>      shl     edx, 2
821      000059AE 01CA    <1>      add      edx, ecx ; PTE offset (for the parent)
822      000059B0 8B1A    <1>      mov      ebx, [edx]
823      000059B2 F6C301 <1>      test     bl, PTE_A_PRESENT ; present or not ?
824      000059B5 7414    <1>      jz       short dapt_2 ; parent process does not use this page
825      000059B7 662500F0 <1>      and      ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
826      000059BB 6681E300F0 <1>      and      bx, PTE_A_CLEAR ; 0F000h ; clear attribute bits
827      000059C0 39D8    <1>      cmp      eax, ebx ; parent's and child's pages are same ?
828      000059C2 7507    <1>      jne      short dapt_2 ; not same page
829      <1>      ; deallocate the child's page
830      000059C4 800A02 <1>      or       byte [edx], PTE_A_WRITE ; convert to writable page (parent)
831      000059C7 59      <1>      pop      ecx
832      000059C8 5B      <1>      pop      ebx
833      000059C9 EB0D    <1>      jmp      short dapt_4
834      <1>
835      <1>      ; 17/04/2021
836      <1>      ; ('dapt_1' is disabled as temporary)
837      <1>      ;
838      <1>      dapt_1:
839      <1>      ; or       eax, eax ; swapped page ?
840      <1>      ; jz       short dapt_5 ; no
841      <1>      ; ; yes
842      <1>      ; shr     eax, 1
843      <1>      ; call    unlink_swap_block ; Deallocate swapped page block
844      <1>      ; ; on the swap disk (or in file)
845      <1>      ; jmp     short dapt_5
846      <1>      dapt_2:
847      000059CB 59      <1>      pop      ecx
848      000059CC 5B      <1>      pop      ebx
849      <1>      dapt_3:
850      <1>      ; 12/07/2016
851      000059CD 66A90004 <1>      test     ax, PTE_SHARED ; shared or direct memory access indicator
852      000059D1 7505    <1>      jnz      short dapt_4 ; AVL bit 1 = 1, do not deallocate this page!
853      <1>      ;
854      <1>      ; and      ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
855      000059D3 E80D000000 <1>      call     deallocate_page ; set the mem allocation bit of this page

```



```

856 <1> dapt_4:
857 <1> ; 20/10/2023
858 <1> ; PTE clearing here was/is not necessary
859 <1> ; because this page table is being deallocated
860 <1> ; and it's PTEs will be ineffective.
861 <1>
862 <1> ;mov dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
863 <1>
864 <1> dapt_1: ; 17/04/2021 (temporary)
865 <1> dapt_5:
866 000059D8 47 <1> inc edi ; page table entry index
867 000059D9 81FF00040000 <1> cmp edi, PAGE_SIZE / 4 ; 1024
868 000059DF 72A6 <1> jnb short dapt_0
869 <1> ;
870 000059E1 58 <1> pop eax ; *
871 000059E2 5A <1> pop edx
872 000059E3 5F <1> pop edi
873 000059E4 5E <1> pop esi
874 <1> ;
875 <1> ;call deallocate_page ; deallocate the page table's itself
876 <1> ;retn
877 <1>
878 <1> deallocate_page:
879 <1> ; 15/09/2015
880 <1> ; 28/04/2015
881 <1> ; 10/03/2015
882 <1> ; 17/10/2014
883 <1> ; 12/10/2014 (Retro UNIX 386 v1 - beginning)
884 <1>
885 <1> ; INPUT ->
886 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
887 <1> ; OUTPUT ->
888 <1> ; [free_pages] is increased
889 <1> ; (corresponding MEMORY ALLOCATION TABLE bit is SET)
890 <1> ; CF = 1 if the page is already deallocated
891 <1> ; (or not allocated) before.
892 <1> ;
893 <1> ; Modified Registers -> EAX
894 <1> ;
895 000059E5 53 <1> push ebx
896 000059E6 52 <1> push edx
897 <1> ;
898 000059E7 C1E80C <1> shr eax, PAGE_SHIFT ; shift physical address to
899 <1> ; 12 bits right
900 <1> ; to get page number
901 000059EA 89C2 <1> mov edx, eax
902 <1> ; 15/09/2015
903 000059EC C1EA03 <1> shr edx, 3 ; to get offset to M.A.T.
904 <1> ; (1 allocation bit = 1 page)
905 <1> ; (1 allocation bytes = 8 pages)
906 000059EF 80E2FC <1> and dl, 0Fch ; clear lower 2 bits
907 <1> ; (to get 32 bit position)
908 <1> ;
909 000059F2 8B00001000 <1> mov ebx, MEM_ALLOC_TBL ; Memory Allocation Table address
910 000059F7 01D3 <1> add ebx, edx
911 000059F9 83E01F <1> and eax, 1Fh ; lower 5 bits only
912 <1> ; (allocation bit position)
913 000059FC 3B15[AC760100] <1> cmp edx, [next_page] ; is the new free page address lower
914 <1> ; than the address in 'next_page' ?
915 <1> ; (next/first free page value)
916 00005A02 7306 <1> jnb short dap_1 ; no
917 00005A04 8915[AC760100] <1> mov [next_page], edx ; yes
918 <1> dap_1:
919 00005A0A 0FAB03 <1> bts [ebx], eax ; unlink/release/deallocate page
920 <1> ; set relevant bit to 1.
921 <1> ; set CF to the previous bit value
922 <1> ; complement carry flag
923 <1> ;cmc short dap_2 ; do not increase free_pages count
924 <1> ;jc ; if the page is already deallocated
925 <1> ; before.
926 00005A0D FF05[A8760100] <1> inc dword [free_pages]
927 <1> dap_2:
928 00005A13 5A <1> pop edx
929 00005A14 5B <1> pop ebx
930 00005A15 C3 <1> ret
931 <1>
932 <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
933 <1> ;
934 <1> ; Copyright (C) KolibriOS team 2004-2012. All rights reserved. ;
935 <1> ; Distributed under terms of the GNU General Public License ;
936 <1> ;
937 <1> ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
938 <1>
939 <1> ;;$Revision: 5057 $
940 <1>
941 <1>
942 <1> ;;align 4
943 <1> ;;proc alloc_page
944 <1>
945 <1> ; pushfd
946 <1> ; cli
947 <1> ; push ebx
948 <1> ;
949 <1> ;;;-
950 <1> ; cmp [pg_data.pages_free], 1
951 <1> ; jle .out_of_memory
952 <1> ;;;-
953 <1> ;
954 <1> ; mov ebx, [page_start]
955 <1> ; mov ecx, [page_end]
956 <1> ;.l1:
957 <1> ; bsf eax, [ebx];
958 <1> ; jnz .found
959 <1> ; add ebx, 4
960 <1> ; cmp ebx, ecx
961 <1> ; jb .l1
962 <1> ; pop ebx
963 <1> ; popfd
964 <1> ; xor eax, eax
965 <1> ; ret
966 <1> ;.found:
967 <1> ;;;-
968 <1> ; dec [pg_data.pages_free]
969 <1> ; jz .out_of_memory
970 <1> ;;;-
971 <1> ; btr [ebx], eax
972 <1> ; mov [page_start], ebx
973 <1> ; sub ebx, sys_pgmap
974 <1> ; lea eax, [eax+ebx*8]
975 <1> ; shl eax, 12
976 <1> ;;;- dec [pg_data.pages_free]
977 <1> ; pop ebx
978 <1> ; popfd
979 <1> ; ret
980 <1> ;;;-

```

```

980      <1> ;;.out_of_memory:
981      <1> ;;      mov     [pg_data.pages_free], 1
982      <1> ;;      xor     eax, eax
983      <1> ;;      pop     ebx
984      <1> ;;      popfd
985      <1> ;;      ret
986      <1> ;;; //-
987      <1> ;;endp
988      <1>
989      <1> duplicate_page_dir:
990      <1>      ; 21/09/2015
991      <1>      ; 31/08/2015
992      <1>      ; 20/07/2015
993      <1>      ; 28/04/2015
994      <1>      ; 27/04/2015
995      <1>      ; 18/04/2015
996      <1>      ; 12/04/2015
997      <1>      ; 18/10/2014
998      <1>      ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
999      <1>
1000     <1>      INPUT ->
1001     <1>      [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
1002     <1>      page directory.
1003     <1>      OUTPUT ->
1004     <1>      EAX = PHYSICAL (real/flat) ADDRESS of the child's
1005     <1>      page directory.
1006     <1>      (New page directory with new page table entries.)
1007     <1>      (New page tables with read only copies of the parent's
1008     <1>      pages.)
1009     <1>      EAX = 0 -> Error (CF = 1)
1010     <1>
1011     <1>      Modified Registers -> none (except EAX)
1012     <1>
1013     <1>      call    allocate_page
1014     <1>      jc      short dpd_err
1015     <1>
1016     <1>      push    ebp ; 20/07/2015
1017     <1>      push    esi
1018     <1>      push    edi
1019     <1>      push    ebx
1020     <1>      push    ecx
1021     <1>      mov     esi, [u.pgdir]
1022     <1>      mov     edi, eax
1023     <1>      push    eax ; save child's page directory address
1024     <1>      ; 31/08/2015
1025     <1>      ; copy PDE 0 from the parent's page dir to the child's page dir
1026     <1>      ; (use same system space for all user page tables)
1027     <1>      movsd   ebp, 1024*4096 ; pass the 1st 4MB (system space)
1028     <1>      mov     ecx, (PAGE_SIZE / 4) - 1 ; 1023
1029     <1>
1030     <1> dpd_0:
1031     <1>      lodsd
1032     <1>      or      eax, eax
1033     <1>      ;jnz     short dpd_1
1034     <1>      test    al, PDE_A_PRESENT ; bit 0 = 1
1035     <1>      jnz     short dpd_1
1036     <1>      ; 20/07/2015 (virtual address at the end of the page table)
1037     <1>      add     ebp, 1024*4096 ; page size * PTE count
1038     <1>      jmp     short dpd_2
1039     <1>
1040     <1> dpd_1:
1041     <1>      and     ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
1042     <1>      mov     ebx, eax
1043     <1>      ; EBX = Parent's page table address
1044     <1>      call    duplicate_page_table
1045     <1>      jc      short dpd_p_err
1046     <1>      ; EAX = Child's page table address
1047     <1>      or      al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
1048     <1>      ; set bit 0, bit 1 and bit 2 to 1
1049     <1>      ; (present, writable, user)
1050     <1>
1051     <1> dpd_2:
1052     <1>      stosd
1053     <1>      loop    dpd_0
1054     <1>
1055     <1>      pop     eax ; restore child's page directory address
1056     <1>
1057     <1> dpd_3:
1058     <1>      pop     ecx
1059     <1>      pop     ebx
1060     <1>      pop     edi
1061     <1>      pop     esi
1062     <1>      pop     ebp ; 20/07/2015
1063     <1>
1064     <1> dpd_err:
1065     <1>      retn
1066     <1>
1067     <1> dpd_p_err:
1068     <1>      ; release the allocated pages missing (recover free space)
1069     <1>      pop     eax ; the new page directory address (physical)
1070     <1>      mov     ebx, [u.pgdir] ; parent's page directory address
1071     <1>      call    deallocate_page_dir
1072     <1>      sub     eax, eax ; 0
1073     <1>      stc
1074     <1>      jmp     short dpd_3
1075     <1>
1076     <1> duplicate_page_table:
1077     <1>      ; 17/04/2021 - TRDOS 386 v2.0.4
1078     <1>      ; (temporary modifications)
1079     <1>      ; 20/02/2017
1080     <1>      ; 21/09/2015
1081     <1>      ; 20/07/2015
1082     <1>      ; 05/05/2015
1083     <1>      ; 28/04/2015
1084     <1>      ; 27/04/2015
1085     <1>      ; 18/04/2015
1086     <1>      ; 18/10/2014
1087     <1>      ; 16/10/2014 (Retro UNIX 386 v1 - beginning)
1088     <1>
1089     <1>      INPUT ->
1090     <1>      ; EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
1091     <1>      ; 20/02/2017
1092     <1>      ; EBP = Linear address of the page (from 'duplicate_page_dir')
1093     <1>      ; (Linear address = CORE + user's virtual address)
1094     <1>      OUTPUT ->
1095     <1>      EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
1096     <1>      (with 'read only' attribute of page table entries)
1097     <1>      ; 20/02/2017
1098     <1>      ; EBP = Next linear page address (for 'duplicate_page_dir')
1099     <1>
1100     <1>      CF = 1 -> error
1101     <1>
1102     <1>      Modified Registers -> EBP (except EAX)
1103     <1>
1104     <1>      call    allocate_page
1105     <1>      jc      short dpt_err
1106     <1>
1107     <1>      push    eax ; *
1108     <1>      push    esi

```

```

1104 00005A76 57      <1>      push    edi
1105 00005A77 52      <1>      push    edx
1106 00005A78 51      <1>      push    ecx
1107                <1>      ;
1108 00005A79 89DE     <1>      mov     esi, ebx
1109 00005A7B 89C7     <1>      mov     edi, eax
1110 00005A7D 89C2     <1>      mov     edx, eax
1111 00005A7F 81C200100000 <1>      add     edx, PAGE_SIZE
1112                <1>      dpt_0:
1113 00005A85 AD      <1>      lodsd
1114 00005A86 21C0     <1>      and     eax, eax
1115 00005A88 7432     <1>      jz      short dpt_3
1116 00005A8A A801     <1>      test    al, PTE_A_PRESENT ; bit 0 = 1
1117                <1>      ; 17/04/2021 (temporary)
1118                <1>      ;jnz    short dpt_1
1119 00005A8C 7439     <1>      jz      short dpt_p_err
1120                <1>      ;
1121                <1>      ; 17/04/2021
1122                <1>      ; ('reload_page' procedure call is disabled as temporary)
1123                <1>      ;
1124                <1>      ; 20/07/2015
1125                <1>      ; ebp = virtual (linear) address of the memory page
1126                <1>      ; call    reload_page ; 28/04/2015
1127                <1>      ;jc      short dpt_p_err
1128                <1>      dpt_1:
1129                <1>      ; 21/09/2015
1130 00005A8E 89C1     <1>      mov     ecx, eax
1131 00005A90 662500F0 <1>      and     ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
1132 00005A94 F6C102 <1>      test    cl, PTE_A_WRITE ; writable page ?
1133 00005A97 751A     <1>      jnz     short dpt_2
1134                <1>      ; Read only (parent) page
1135                <1>      ; - there is a third process which uses this page -
1136                <1>      ; Allocate a new page for the child process
1137 00005A99 E888FDFFFF <1>      call    allocate_page
1138 00005A9E 7227     <1>      jc      short dpt_p_err
1139 00005AA0 57      <1>      push    edi
1140 00005AA1 56      <1>      push    esi
1141 00005AA2 89CE     <1>      mov     esi, ecx
1142 00005AA4 89C7     <1>      mov     edi, eax
1143 00005AA6 B900040000 <1>      mov     ecx, PAGE_SIZE/4
1144 00005AAB F3A5     <1>      rep     movsd ; copy page (4096 bytes)
1145 00005AAD 5E      <1>      pop     esi
1146 00005AAE 5F      <1>      pop     edi
1147                <1>      ;
1148                <1>      ;
1149                <1>      ; 17/04/2021
1150                <1>      ; ('add_to_swap_queue' procedure call is disabled as temporary)
1151                <1>      ;
1152                <1>      ; push    ebx
1153                <1>      ; push    eax
1154                <1>      ; 20/07/2015
1155                <1>      ; mov     ebx, ebp
1156                <1>      ; ebp = virtual (linear) address of the memory page
1157                <1>      ; call    add_to_swap_queue
1158                <1>      ; pop     eax
1159                <1>      ; pop     ebx
1160                <1>      ;
1161                <1>      ; 21/09/2015
1162 00005AAF 0C07     <1>      or      al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
1163                <1>      ; user + writable + present page
1164 00005AB1 EB09     <1>      jmp     short dpt_3
1165                <1>      dpt_2:
1166                <1>      ;or      ax, PTE_A_USER+PTE_A_PRESENT
1167 00005AB3 0C05     <1>      or      al, PTE_A_USER+PTE_A_PRESENT
1168                <1>      ; (read only page!)
1169 00005AB5 8946FC <1>      mov     [esi-4], eax ; update parent's PTE
1170 00005AB8 660D0002 <1>      or      ax, PTE_DUPLICATED ; (read only page & duplicated PTE!)
1171                <1>      dpt_3:
1172 00005ABC AB      <1>      stosd   ; EDI points to child's PTE
1173                <1>      ;
1174 00005ABD 81C500100000 <1>      add     ebp, 4096 ; 20/07/2015 (next page)
1175                <1>      ;
1176 00005AC3 39D7     <1>      cmp     edi, edx
1177 00005AC5 72BE     <1>      jb      short dpt_0
1178                <1>      dpt_p_err:
1179                <1>      pop     ecx
1180 00005AC8 5A      <1>      pop     edx
1181 00005AC9 5F      <1>      pop     edi
1182 00005ACA 5E      <1>      pop     esi
1183 00005ACB 58      <1>      pop     eax ; *
1184                <1>      dpt_err:
1185 00005ACC C3      <1>      retn
1186                <1>      ;
1187                <1>      page_fault_handler: ; CPU EXCEPTION 0Eh (14) : Page Fault !
1188                <1>      ; 29/11/2023 - TRDOS 386 v2.0.7
1189                <1>      ; 17/04/2021 - TRDOS 386 v2.0.4
1190                <1>      ; (temporary modifications)
1191                <1>      ; 21/09/2015
1192                <1>      ; 19/09/2015
1193                <1>      ; 17/09/2015
1194                <1>      ; 28/08/2015
1195                <1>      ; 20/07/2015
1196                <1>      ; 28/06/2015
1197                <1>      ; 03/05/2015
1198                <1>      ; 30/04/2015
1199                <1>      ; 18/04/2015
1200                <1>      ; 12/04/2015
1201                <1>      ; 30/10/2014
1202                <1>      ; 11/09/2014
1203                <1>      ; 10/09/2014 (Retro UNIX 386 v1 - beginning)
1204                <1>      ;
1205                <1>      ; Note: This is not an interrupt/exception handler.
1206                <1>      ; This is a 'page fault remedy' subroutine
1207                <1>      ; which will be called by standard/uniform
1208                <1>      ; exception handler.
1209                <1>      ;
1210                <1>      ; INPUT ->
1211                <1>      ; [error_code] = 32 bit ERROR CODE (lower 5 bits are valid)
1212                <1>      ;
1213                <1>      ; cr2 = the virtual (linear) address
1214                <1>      ; which has caused to page fault (19/09/2015)
1215                <1>      ;
1216                <1>      ; OUTPUT ->
1217                <1>      ; (Corresponding PAGE TABLE ENTRY is mapped/set)
1218                <1>      ; EAX = 0 -> no error
1219                <1>      ; EAX > 0 -> error code in EAX (also CF = 1)
1220                <1>      ;
1221                <1>      ; Modified Registers -> none (except EAX)
1222                <1>      ;
1223                <1>      ;
1224                <1>      ; ERROR CODE:
1225                <1>      ; 31 ..... 4 3 2 1 0
1226                <1>      ; +---+---+---+---+---+---+---+---+
1227                <1>      ; | Reserved | I | R | U | W | P |

```

```

1228      ; P : PRESENT - When set, the page fault was caused by
1229      ;               a page-protection violation. When not set,
1230      ;               it was caused by a non-present page.
1231      ; W : WRITE  - When set, the page fault was caused by
1232      ;               a page write. When not set, it was caused
1233      ;               by a page read.
1234      ; U : USER   - When set, the page fault was caused
1235      ;               while CPL = 3.
1236      ;               This does not necessarily mean that
1237      ;               the page fault was a privilege violation.
1238      ; R : RESERVD - When set, the page fault was caused by
1239      ;               writing a 1 in a reserved field.
1240      ; I : INSTRUC - When set, the page fault was caused by
1241      ;               an instruction fetch
1242      ;
1243      ; x86 (32 bit) VIRTUAL ADDRESS TRANSLATION
1244      ;
1245      ; 31          22          12 11          0
1246      ; +-----+-----+-----+-----+
1247      ; | PAGE DIR. ENTRY # | PAGE TAB. ENTRY # | OFFSET |
1248      ; +-----+-----+-----+-----+
1249      ;
1250      ;
1251      ; CR3 REGISTER (Control Register 3)
1252      ;
1253      ; 31          12          5 4 3 2 0
1254      ; +-----+-----+-----+-----+
1255      ; | PAGE DIRECTORY TABLE BASE ADDRESS | reserved | P | P |
1256      ; |                                     |          | C | W | rsvrd |
1257      ; |                                     |          | D | T |
1258      ; +-----+-----+-----+-----+
1259      ;
1260      ; PWT - WRITE THROUGH
1261      ; PCD - CACHE DISABLE
1262      ;
1263      ;
1264      ; x86 PAGE DIRECTORY ENTRY (4 KByte Page)
1265      ; 31          12 11 9 8 7 6 5 4 3 2 1 0
1266      ; +-----+-----+-----+-----+
1267      ; | PAGE TABLE BASE ADDRESS 31..12 | AVL | G | 0 | D | A | C | W | / | / | P | U | R |
1268      ; |                                     |     |   |   |   |   |   |   |   |   |   |   |   |
1269      ; |                                     |     |   |   |   |   |   |   |   |   |   |   |   |
1270      ; +-----+-----+-----+-----+
1271      ;
1272      ; P - PRESENT
1273      ; R/W - READ/WRITE
1274      ; U/S - USER/SUPERVISOR
1275      ; PWT - WRITE THROUGH
1276      ; PCD - CACHE DISABLE
1277      ; A - ACCESSED
1278      ; D - DIRTY (IGNORED)
1279      ; PAT - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
1280      ; G - GLOBAL (IGNORED)
1281      ; AVL - AVAILABLE FOR SYSTEMS PROGRAMMER USE
1282      ;
1283      ;
1284      ; x86 PAGE TABLE ENTRY (4 KByte Page)
1285      ; 31          12 11 9 8 7 6 5 4 3 2 1 0
1286      ; +-----+-----+-----+-----+
1287      ; | PAGE FRAME BASE ADDRESS 31..12 | AVL | G | P | A | D | A | C | W | / | / | P | U | R |
1288      ; |                                     |     |   |   |   |   |   |   |   |   |   |   |   |   |
1289      ; |                                     |     |   |   |   |   |   |   |   |   |   |   |   |   |
1290      ; +-----+-----+-----+-----+
1291      ;
1292      ; P - PRESENT
1293      ; R/W - READ/WRITE
1294      ; U/S - USER/SUPERVISOR
1295      ; PWT - WRITE THROUGH
1296      ; PCD - CACHE DISABLE
1297      ; A - ACCESSED
1298      ; D - DIRTY
1299      ; PAT - PAGE ATTRIBUTE TABLE INDEX (CACHE BEHAVIOR)
1300      ; G - GLOBAL
1301      ; AVL - AVAILABLE FOR SYSTEMS PROGRAMMER USE
1302      ;
1303      ;
1304      ; 80386 PAGE TABLE ENTRY (4 KByte Page)
1305      ; 31          12 11 9 8 7 6 5 4 3 2 1 0
1306      ; +-----+-----+-----+-----+
1307      ; | PAGE FRAME BASE ADDRESS 31..12 | AVL | 0 | 0 | D | A | 0 | 0 | U | R |
1308      ; |                                     |     |   |   |   |   |   |   |   |   |   |
1309      ; |                                     |     |   |   |   |   |   |   |   |   |   |   |
1310      ; +-----+-----+-----+-----+
1311      ;
1312      ; P - PRESENT
1313      ; R/W - READ/WRITE
1314      ; U/S - USER/SUPERVISOR
1315      ; D - DIRTY
1316      ; AVL - AVAILABLE FOR SYSTEMS PROGRAMMER USE
1317      ;
1318      ; NOTE: 0 INDICATES INTEL RESERVED. DO NOT DEFINE.
1319      ;
1320      ;
1321      ; Invalid Page Table Entry
1322      ; 31          1 0
1323      ; +-----+-----+-----+-----+
1324      ; | AVAILABLE | 0 |
1325      ; +-----+-----+-----+-----+
1326      ;
1327      ;
1328      ;
1329      ;
1330      00005ACD 53      push    ebx
1331      00005ACE 52      push    edx
1332      00005ACF 51      push    ecx
1333      ;
1334      ; 21/09/2015 (debugging)
1335      00005AD0 FF05[C48E0100] inc     dword [u.pfcount] ; page fault count for running process
1336      00005AD6 FF05[70900100] inc     dword [PF_Count] ; total page fault count
1337      ; 28/06/2015
1338      ; mov     edx, [error_code] ; Lower 5 bits are valid
1339      00005ADC 8A15[68900100] mov     dl, [error_code]
1340      ;
1341      00005AE2 F6C201      test    dl, 1 ; page fault was caused by a non-present page
1342      ; sign
1343      00005AE5 7425      jz      short pfh_alloc_np
1344      ;
1345      ; If it is not a 'write on read only page' type page fault
1346      ; major page fault error with minor reason must be returned without
1347      ; fixing the problem. 'sys_exit with error' will be needed
1348      ; after return here!
1349      ; Page fault will be remedied, by copying page contents
1350      ; to newly allocated page with write permission;
1351      ; sys_fork -> sys_exec -> copy on write, demand paging method is

```

```

1352 <1> ; used for working with minimum possible memory usage.
1353 <1> ; sys_fork will duplicate page directory and tables of parent
1354 <1> ; process with 'read only' flag. If the child process attempts to
1355 <1> ; write on these read only pages, page fault will be directed here
1356 <1> ; for allocating a new page with same data/content.
1357 <1> ;
1358 <1> ; IMPORTANT : Retro UNIX 386 v1 (and SINGLIX and TR-DOS)
1359 <1> ; will not force to separate CODE and DATA space
1360 <1> ; in a process/program...
1361 <1> ; CODE segment/section may contain DATA!
1362 <1> ; It is flat, smoth and simplest programming method already as in
1363 <1> ; Retro UNIX 8086 v1 and MS-DOS programs.
1364 <1> ;
1365 00005AE7 F6C202 <1> test dl, 2 ; page fault was caused by a page write
1366 <1> ; sign
1367 00005AEA 7418 <1> jz short pfh_p_err
1368 <1> ; 31/08/2015
1369 00005AEC F6C204 <1> test dl, 4 ; page fault was caused while CPL = 3 (user mode)
1370 <1> ; sign. (U+W+P = 4+2+1 = 7)
1371 00005AEF 7413 <1> jz short pfh_pv_err
1372 <1> ;
1373 <1> ; make a new page and copy the parent's page content
1374 <1> ; as the child's new page content
1375 <1> ;
1376 00005AF1 0F20D3 <1> mov ebx, cr2 ; CR2 contains the linear address
1377 <1> ; which has caused to page fault
1378 00005AF4 E87C000000 <1> call copy_page
1379 00005AF9 7202 <1> jc short pfh_im_err ; insufficient memory
1380 <1> ;
1381 00005AFB EB72 <1> jmp pfh_cpp_ok
1382 <1> ;
1383 <1> ; 29/11/2023
1384 <1> pfh_im_err:
1385 00005AFD B8E4000000 <1> mov eax, ERR_MAJOR_PF + ERR_MINOR_IM ; Error code in AX
1386 <1> ; Major (Primary) Error: Page Fault
1387 <1> ; Minor (Secondary) Error: Insufficient Memory !
1388 00005B02 EB6D <1> jmp short pfh_err_retn
1389 <1> ;
1390 <1> ; 29/11/2023
1391 <1> pfh_p_err: ; 09/03/2015
1392 <1> pfh_pv_err:
1393 <1> ; Page fault was caused by a protection-violation
1394 00005B04 B8E6000000 <1> mov eax, ERR_MAJOR_PF + ERR_MINOR_PV ; Error code in AX
1395 <1> ; Major (Primary) Error: Page Fault
1396 <1> ; Minor (Secondary) Error: Protection violation !
1397 00005B09 F9 <1> stc
1398 00005B0A EB65 <1> jmp short pfh_err_retn
1399 <1> ;
1400 <1> pfh_alloc_np:
1401 00005B0C E815FDFFFF <1> call allocate_page ; (allocate a new page)
1402 00005B11 72EA <1> jc pfh_im_err ; 'insufficient memory' error
1403 <1> pfh_chk_cpl:
1404 <1> ; EAX = Physical (base) address of the allocated (new) page
1405 <1> ; (Lower 12 bits are ZERO, because
1406 <1> ; the address is on a page boundary)
1407 00005B13 80E204 <1> and dl, 4 ; CPL = 3 ?
1408 00005B16 7505 <1> jnz short pfh_um
1409 <1> ; Page fault handler for kernel/system mode (CPL=0)
1410 00005B18 0F20DB <1> mov ebx, cr3 ; CR3 (Control Register 3) contains physical address
1411 <1> ; of the current/active page directory
1412 <1> ; (Always kernel/system mode page directory, here!)
1413 <1> ; Note: Lower 12 bits are 0. (page boundary)
1414 00005B1B EB06 <1> jmp short pfh_get_pde
1415 <1> ;
1416 <1> pfh_um: ; Page fault handler for user/appl. mode (CPL=3)
1417 00005B1D 8B1D[AC8E0100] <1> mov ebx, [u.pgdir] ; Page directory of current/active process
1418 <1> ; Physical address of the USER's page directory
1419 <1> ; Note: Lower 12 bits are 0. (page boundary)
1420 <1> pfh_get_pde:
1421 00005B23 80CA03 <1> or dl, 3 ; USER + WRITE + PRESENT or SYSTEM + WRITE + PRESENT
1422 00005B26 0F20D1 <1> mov ecx, cr2 ; CR2 contains the virtual address
1423 <1> ; which has been caused to page fault
1424 <1> ;
1425 00005B29 C1E914 <1> shr ecx, 20 ; shift 20 bits right
1426 00005B2C 80E1FC <1> and cl, 0FCh ; mask lower 2 bits to get PDE offset
1427 <1> ;
1428 00005B2F 01CB <1> add ebx, ecx ; now, EBX points to the relevant page dir entry
1429 00005B31 880B <1> mov ecx, [ebx] ; physical (base) address of the page table
1430 00005B33 F6C101 <1> test cl, 1 ; check bit 0 is set (1) or not (0).
1431 00005B36 740B <1> jz short pfh_set_pde ; Page directory entry is not valid,
1432 <1> ; set/validate page directory entry
1433 00005B38 6681E100F0 <1> and cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
1434 00005B3D 89CB <1> mov ebx, ecx ; Physical address of the page table
1435 00005B3F 89C1 <1> mov ecx, eax ; new page address (physical)
1436 00005B41 EB16 <1> jmp short pfh_get_pte
1437 <1> pfh_set_pde:
1438 <1> ; NOTE: Page directories and page tables never be swapped out!
1439 <1> ; (So, we know this PDE is empty or invalid)
1440 <1> ;
1441 00005B43 08D0 <1> or al, dl ; lower 3 bits are used as U/S, R/W, P flags
1442 00005B45 8903 <1> mov [ebx], eax ; Let's put the new page directory entry here !
1443 00005B47 30C0 <1> xor al, al ; clear lower (3..8) bits
1444 00005B49 89C3 <1> mov ebx, eax
1445 00005B4B E8D6FCFFFF <1> call allocate_page ; (allocate a new page)
1446 00005B50 72AB <1> jc short pfh_im_err ; 'insufficient memory' error
1447 <1> pfh_spde_1:
1448 <1> ; EAX = Physical (base) address of the allocated (new) page
1449 00005B52 89C1 <1> mov ecx, eax
1450 00005B54 E83EFDFFFF <1> call clear_page ; Clear page content
1451 <1> pfh_get_pte:
1452 00005B59 0F20D0 <1> mov eax, cr2 ; virtual address
1453 <1> ; which has been caused to page fault
1454 00005B5C 89C7 <1> mov edi, eax ; 20/07/2015
1455 00005B5E C1E80C <1> shr eax, 12 ; shift 12 bit right to get
1456 <1> ; higher 20 bits of the page fault address
1457 00005B61 25FF030000 <1> and eax, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
1458 00005B66 C1E002 <1> shl eax, 2 ; shift 2 bits left to get PTE offset
1459 00005B69 01C3 <1> add ebx, eax ; now, EBX points to the relevant page table entry
1460 <1> ; 17/04/2021 temporary
1461 <1> ; mov eax, [ebx] ; get previous value of pte
1462 <1> ; ; bit 0 of EAX is always 0 (otherwise we would not be here)
1463 <1> ;
1464 <1> ; 17/04/2021
1465 <1> ; ('swap_in' procedure call has been disabled as temporary)
1466 <1> ;
1467 <1> ; and eax, eax
1468 <1> ; jz short pfh_gpte_1
1469 <1> ; ; 20/07/2015
1470 <1> ; xchg ebx, ecx ; new page address (physical)
1471 <1> ; push ebp ; 20/07/2015
1472 <1> ; mov ebp, cr2
1473 <1> ; ; ECX = physical address of the page table entry
1474 <1> ; ; EBX = Memory page address (physical!)
1475 <1> ; ; EAX = Swap disk (offset) address

```

```

1476 <1> ; ; EBP = virtual address (page fault address)
1477 <1> ; call swap_in
1478 <1> ; pop ebp
1479 <1> ; jc short pfh_err_retn
1480 <1> ; xchg ecx, ebx
1481 <1> ; ; EBX = physical address of the page table entry
1482 <1> ; ; ECX = new page
1483 <1> pfh_gpte_1:
1484 <1> or cl, dl ; lower 3 bits are used as U/S, R/W, P flags
1485 <1> mov [ebx], ecx ; Let's put the new page table entry here !
1486 <1> pfh_cpp_ok:
1487 <1> ; 17/04/2021
1488 <1> ; ('add_to_swap_queue' procedure call has been disabled as temporary)
1489 <1> ;
1490 <1> ; ; 20/07/2015
1491 <1> ; mov ebx, cr2
1492 <1> ; call add_to_swap_queue
1493 <1> ;
1494 <1> ; The new PTE (which contains the new page) will be added to
1495 <1> ; the swap queue, here.
1496 <1> ; (Later, if memory will become insufficient,
1497 <1> ; one page will be swapped out which is at the head of
1498 <1> ; the swap queue by using FIFO and access check methods.)
1499 <1> ;
1500 <1> xor eax, eax ; 0
1501 <1> ;
1502 <1> pfh_err_retn:
1503 <1> pop ecx
1504 <1> pop edx
1505 <1> pop ebx
1506 <1> retn
1507 <1> ;
1508 <1> copy_page:
1509 <1> ; 29/08/2023 (TRDOS 386 v2.0.6)
1510 <1> ; 22/09/2015
1511 <1> ; 21/09/2015
1512 <1> ; 19/09/2015
1513 <1> ; 07/09/2015
1514 <1> ; 31/08/2015
1515 <1> ; 20/07/2015
1516 <1> ; 05/05/2015
1517 <1> ; 03/05/2015
1518 <1> ; 18/04/2015
1519 <1> ; 12/04/2015
1520 <1> ; 30/10/2014
1521 <1> ; 18/10/2014 (Retro UNIX 386 v1 - beginning)
1522 <1> ;
1523 <1> ; INPUT ->
1524 <1> ; EBX = virtual (linear) address of source page
1525 <1> ; (Page fault address)
1526 <1> ; OUTPUT ->
1527 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF THE ALLOCATED PAGE
1528 <1> ; (corresponding PAGE TABLE ENTRY is mapped/set)
1529 <1> ; EAX = 0 (CF = 1)
1530 <1> ; if there is not a free page to be allocated
1531 <1> ; (page content of the source page will be copied
1532 <1> ; onto the target/new page)
1533 <1> ;
1534 <1> ; Modified Registers -> ecx, ebx (except EAX)
1535 <1> ;
1536 <1> push esi
1537 <1> push edi
1538 <1> ;push ebx
1539 <1> ;push ecx
1540 <1> xor esi, esi
1541 <1> shr ebx, 12 ; shift 12 bits right to get PDE & PTE numbers
1542 <1> mov ecx, ebx ; save page fault address (as 12 bit shifted)
1543 <1> shr ebx, 8 ; shift 8 bits right and then
1544 <1> and bl, 0Fch ; mask lower 2 bits to get PDE offset
1545 <1> mov edi, ebx ; save it for the parent of current process
1546 <1> add ebx, [u.pgdir] ; EBX points to the relevant page dir entry
1547 <1> mov eax, [ebx] ; physical (base) address of the page table
1548 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
1549 <1> mov ebx, ecx ; (restore higher 20 bits of page fault address)
1550 <1> and ebx, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
1551 <1> shl bx, 2 ; shift 2 bits left to get PTE offset
1552 <1> ; 29/08/2023
1553 <1> shl ebx, 2
1554 <1> add ebx, eax ; EBX points to the relevant page table entry
1555 <1> ; 07/09/2015
1556 <1> test word [ebx], PTE_DUPLICATED ; (Does current process share this
1557 <1> ; ; read only page as a child process?)
1558 <1> jnz short cpp_0 ; yes
1559 <1> mov ecx, [ebx] ; PTE value
1560 <1> and cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
1561 <1> jmp short cpp_1
1562 <1> cpp_0:
1563 <1> mov esi, edi
1564 <1> add esi, [u.pgdir] ; the parent's page directory entry
1565 <1> mov eax, [esi] ; physical (base) address of the page table
1566 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
1567 <1> mov esi, ecx ; (restore higher 20 bits of page fault address)
1568 <1> and esi, 3FFh ; mask PDE# bits, the result is PTE# (0 to 1023)
1569 <1> shl si, 2 ; shift 2 bits left to get PTE offset
1570 <1> ; 29/08/2023
1571 <1> shl esi, 2
1572 <1> add esi, eax ; EDX points to the relevant page table entry
1573 <1> mov ecx, [esi] ; PTE value of the parent process
1574 <1> ; 21/09/2015
1575 <1> mov eax, [ebx] ; PTE value of the child process
1576 <1> and ax, PTE_A_CLEAR ; 0F000h ; clear page attributes
1577 <1> ;
1578 <1> test cl, PTE_A_PRESENT ; is it a present/valid page ?
1579 <1> jz short cpp_3 ; the parent's page is not same page
1580 <1> ;
1581 <1> and cx, PTE_A_CLEAR ; 0F000h ; clear page attributes
1582 <1> cmp eax, ecx ; Same page?
1583 <1> jne short cpp_3 ; Parent page and child page are not same
1584 <1> ; Convert child's page to writable page
1585 <1> ;
1586 <1> cpp_1:
1587 <1> call allocate_page
1588 <1> jc short cpp_4 ; 'insufficient memory' error
1589 <1> and esi, esi ; check ESI is valid or not
1590 <1> jz short cpp_2
1591 <1> ; Convert read only page to writable page
1592 <1> ; (for the parent of the current process)
1593 <1> ; and word [esi], PTE_A_CLEAR ; 0F000h
1594 <1> ; 22/09/2015
1595 <1> mov [esi], ecx
1596 <1> or byte [esi], PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER
1597 <1> ; 1+2+4 = 7
1598 <1> cpp_2:
1599 <1> mov edi, eax ; new page address of the child process
1600 <1> ; 07/09/2015

```

```

1600 00005BF2 89CE      <1>      mov     esi, ecx ; the page address of the parent process
1601 00005BF4 B900040000 <1>      mov     ecx, PAGE_SIZE / 4
1602 00005BF9 F3A5      <1>      rep     movsd ; 31/08/2015
1603                  <1>      cpp_3:
1604 00005BFB 0C07      <1>      or      al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 1+2+4 = 7
1605 00005BFD 8903      <1>      mov     [ebx], eax ; Update PTE
1606 00005BFF 28C0      <1>      sub     al, al ; clear attributes
1607                  <1>      cpp_4:
1608                  <1>      ;pop     ecx
1609                  <1>      ;pop     ebx
1610 00005C01 5F      <1>      pop     edi
1611 00005C02 5E      <1>      pop     esi
1612 00005C03 C3      <1>      retn
1613                  <1>
1614                  <1>      ; 28/04/2015
1615                  <1>      ; 24/10/2014
1616                  <1>      ; 21/10/2014 (Retro UNIX 386 v1 - beginning)
1617                  <1>      ; SWAP_PAGE_QUEUE (4096 bytes)
1618                  <1>
1619                  <1>      0000    0001    0002    0003    ....    1020    1021    1022    1023
1620                  <1>      +-----+-----+-----+-----+-----+-----+-----+-----+
1621                  <1>      | pg1 | pg2 | pg3 | pg4 | .... |pg1021|pg1022|pg1023|pg1024|
1622                  <1>      +-----+-----+-----+-----+-----+-----+-----+-----+
1623                  <1>
1624                  <1>      [swpq_last] = 0 to 4096 (step 4) -> the last position on the queue
1625                  <1>
1626                  <1>      Method:
1627                  <1>      Swap page queue is a list of allocated pages with physical
1628                  <1>      addresses (system mode virtual addresses = physical addresses).
1629                  <1>      It is used for 'swap_in' and 'swap_out' procedures.
1630                  <1>      When a new page is being allocated, swap queue is updated
1631                  <1>      by 'swap_queue_shift' procedure, header of the queue (offset 0)
1632                  <1>      is checked for 'accessed' flag. If the 1st page on the queue
1633                  <1>      is 'accessed' or 'read only', it is dropped from the list;
1634                  <1>      other pages from the 2nd to the last (in [swpq_last]) shifted
1635                  <1>      to head then the 2nd page becomes the 1st and '[swpq_last]'
1636                  <1>      offset value becomes it's previous offset value - 4.
1637                  <1>      If the 1st page of the swap page queue is not 'accessed'
1638                  <1>      the queue/list is not shifted.
1639                  <1>      After the queue/list shift, newly allocated page is added
1640                  <1>      to the tail of the queue at the [swpq_count*4] position.
1641                  <1>      But, if [swpq_count] > 1023, the newly allocated page
1642                  <1>      will not be added to the tail of swap page queue.
1643                  <1>
1644                  <1>      During 'swap_out' procedure, swap page queue is checked for
1645                  <1>      the first non-accessed, writable page in the list,
1646                  <1>      from the head to the tail. The list is shifted to left
1647                  <1>      (to the head) till a non-accessed page will be found in the list.
1648                  <1>      Then, this page is swapped out (to disk) and then it is dropped
1649                  <1>      from the list by a final swap queue shift. [swpq_count] value
1650                  <1>      is changed. If all pages on the queue are 'accessed',
1651                  <1>      'insufficient memory' error will be returned ('swap_out'
1652                  <1>      procedure will be failed)...
1653                  <1>
1654                  <1>      Note: If the 1st page of the queue is an 'accessed' page,
1655                  <1>      'accessed' flag of the page will be reset (0) and that page
1656                  <1>      (PTE) will be added to the tail of the queue after
1657                  <1>      the check, if [swpq_count] < 1023. If [swpq_count] = 1024
1658                  <1>      the queue will be rotated and the PTE in the head will be
1659                  <1>      added to the tail after resetting 'accessed' bit.
1660                  <1>
1661                  <1>
1662                  <1>
1663                  <1>      SWAP DISK/FILE (with 4096 bytes swapped page blocks)
1664                  <1>
1665                  <1>      00000000 00000004 00000008 0000000c ... size-8 size-4
1666                  <1>      +-----+-----+-----+-----+-----+-----+-----+-----+
1667                  <1>      |descriptr| page(1) | page(2) | page(3) | ... |page(n-1)| page(n) |
1668                  <1>      +-----+-----+-----+-----+-----+-----+-----+-----+
1669                  <1>
1670                  <1>      [swpd_next] = the first free block address in swapped page records
1671                  <1>      for next free block search by 'swap_out' procedure.
1672                  <1>      [swpd_size] = swap disk/file size in sectors (512 bytes)
1673                  <1>      NOTE: max. possible swap disk size is 1024 GB
1674                  <1>      (entire swap space must be accessed by using
1675                  <1>      31 bit offset address)
1676                  <1>      [swpd_free] = free block (4096 bytes) count in swap disk/file space
1677                  <1>      [swpd_start] = absolute/start address of the swap disk/file
1678                  <1>      0 for file, or beginning sector of the swap partition
1679                  <1>      [swp_drv] = logical drive description table addr. of swap disk/file
1680                  <1>
1681                  <1>
1682                  <1>      Method:
1683                  <1>      When the memory (ram) becomes insufficient, page allocation
1684                  <1>      procedure swaps out a page from memory to the swap disk
1685                  <1>      (partition) or swap file to get a new free page at the memory.
1686                  <1>      Swapping out is performed by using swap page queue.
1687                  <1>
1688                  <1>      Allocation block size of swap disk/file is equal to page size
1689                  <1>      (4096 bytes). Swapping address (in sectors) is recorded
1690                  <1>      into relevant page file entry as 31 bit physical (logical)
1691                  <1>      offset address as 1 bit shifted to left for present flag (0).
1692                  <1>      Swapped page address is between 1 and swap disk/file size - 4.
1693                  <1>      Absolute physical (logical) address of the swapped page is
1694                  <1>      calculated by adding offset value to the swap partition's
1695                  <1>      start address. If the swap device (disk) is a virtual disk
1696                  <1>      or it is a file, start address of the swap disk/volume is 0,
1697                  <1>      and offset value is equal to absolute (physical or logical)
1698                  <1>      address/position. (It has not to be ZERO if the swap partition
1699                  <1>      is in a partitioned virtual hard disk.)
1700                  <1>
1701                  <1>
1702                  <1>      Note: Swap addresses are always specified/declared in sectors,
1703                  <1>      not in bytes or in blocks/zones/clusters (4096 bytes) as unit.
1704                  <1>
1705                  <1>      Swap disk/file allocation is mapped via 'Swap Allocation Table'
1706                  <1>      at memory as similar to 'Memory Allocation Table'.
1707                  <1>
1708                  <1>      Every bit of Swap Allocation Table represents one swap block
1709                  <1>      (equal to page size) respectively. Bit 0 of the S.A.T. byte 0
1710                  <1>      is reserved for swap disk/file block 0 as descriptor block
1711                  <1>      (also for compatibility with PTE). If bit value is ZERO,
1712                  <1>      it means relevant (respective) block is in use, and,
1713                  <1>      of course, if bit value is 1, it means relevant (respective)
1714                  <1>      swap disk/file block is free.
1715                  <1>      For example: bit 1 of the byte 128 represents block 1025
1716                  <1>      (128*8+1) or sector (offset) 8200 on the swap disk or
1717                  <1>      byte (offset/position) 4198400 in the swap file.
1718                  <1>      4GB swap space is represented via 128KB Swap Allocation Table.
1719                  <1>      Initial layout of Swap Allocation Table is as follows:
1720                  <1>
1721                  <1>      01111111111111111111111111111111 .... 11111111111111111111111111111111
1722                  <1>      -----
1723                  <1>      (0 is reserved block, 1s represent free blocks respectively.)
                  <1>      (Note: Allocation cell/unit of the table is bit, not byte)

```

```

1724 <1> ;
1725 <1> ;
1726 <1> ; .....
1727 <1> ;
1728 <1> ; 'swap_out' procedure checks 'free_swap_blocks' count at first,
1729 <1> ; then it searches Swap Allocation Table if free count is not
1730 <1> ; zero. From beginning the [swpd_next] dword value, the first bit
1731 <1> ; position with value of 1 on the table is converted to swap
1732 <1> ; disk/file offset address, in sectors (not 4096 bytes block).
1733 <1> ; 'ldrv_write' procedure is called with ldrv (logical drive
1734 <1> ; number of physical swap disk or virtual swap disk)
1735 <1> ; number, sector offset (not absolute sector -LBA- number),
1736 <1> ; and sector count (8, 512*8 = 4096) and buffer address
1737 <1> ; (memory page). That will be a direct disk write procedure.
1738 <1> ; (for preventing late memory allocation, significant waiting).
1739 <1> ; If disk write procedure returns with error or free count of
1740 <1> ; swap blocks is ZERO, 'swap_out' procedure will return with
1741 <1> ; 'insufficient memory error' (cf=1).
1742 <1> ;
1743 <1> ; (Note: Even if free swap disk/file blocks was not zero,
1744 <1> ; any disk write error will not be fixed by 'swap_out' procedure,
1745 <1> ; in other words, 'swap_out' will not check the table for other
1746 <1> ; free blocks after a disk write error. It will return to
1747 <1> ; the caller with error (CF=1) which means swapping is failed.
1748 <1> ;
1749 <1> ; After writing the page on to swap disk/file address/sector,
1750 <1> ; 'swap_out' procedure returns with that swap (offset) sector
1751 <1> ; address (cf=0).
1752 <1> ;
1753 <1> ; .....
1754 <1> ;
1755 <1> ; 'swap_in' procedure loads addressed (relevant) swap disk or
1756 <1> ; file sectors at specified memory page. Then page allocation
1757 <1> ; procedure updates relevant page table entry with 'present'
1758 <1> ; attribute. If swap disk or file reading fails there is nothing
1759 <1> ; to do, except to terminate the process which is the owner of
1760 <1> ; the swapped page.
1761 <1> ;
1762 <1> ; 'swap_in' procedure sets the relevant/respective bit value
1763 <1> ; in the Swap Allocation Table (as free block). 'swap_in' also
1764 <1> ; updates [swpd_first] pointer if it is required.
1765 <1> ;
1766 <1> ; .....
1767 <1> ;
1768 <1> ; Note: If [swap_enabled] value is ZERO, that means there is not
1769 <1> ; a swap disk or swap file in use... 'swap_in' and 'swap_out'
1770 <1> ; procedures and 'swap page que' procedures will not be active...
1771 <1> ; 'Insufficient memory' error will be returned by 'swap_out'
1772 <1> ; and 'general protection fault' will be returned by 'swap_in'
1773 <1> ; procedure, if it is called mistakenly (a wrong value in a PTE).
1774 <1> ;
1775 <1> ; 17/04/2021
1776 <1> ; ('swap_in' procedure call is disabled as temporary)
1777 <1> ;
1778 <1> swap_in:
1779 <1> ; 31/08/2015
1780 <1> ; 20/07/2015
1781 <1> ; 28/04/2015
1782 <1> ; 18/04/2015
1783 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
1784 <1> ;
1785 <1> ; INPUT ->
1786 <1> ; EBX = PHYSICAL (real/flat) ADDRESS OF THE MEMORY PAGE
1787 <1> ; EBP = VIRTUAL (LINEAR) ADDRESS (page fault address)
1788 <1> ; EAX = Offset Address for the swapped page on the
1789 <1> ; swap disk or in the swap file.
1790 <1> ;
1791 <1> ; OUTPUT ->
1792 <1> ; EAX = 0 if loading at memory has been successful
1793 <1> ;
1794 <1> ; CF = 1 -> swap disk reading error (disk/file not present
1795 <1> ; or sector not present or drive not ready
1796 <1> ; EAX = Error code
1797 <1> ; [u.error] = EAX
1798 <1> ; = The last error code for the process
1799 <1> ; (will be reset after returning to user)
1800 <1> ;
1801 <1> ; Modified Registers -> EAX
1802 <1> ;
1803 <1> ;
1804 <1> ; cmp dword [swp_drv], 0
1805 <1> ; jna short swpin_dnp_err
1806 <1> ;
1807 <1> ; cmp eax, [swpd_size]
1808 <1> ; jnb short swpin_snp_err
1809 <1> ;
1810 <1> ; push esi
1811 <1> ; push ebx
1812 <1> ; push ecx
1813 <1> ; mov esi, [swp_drv]
1814 <1> ; mov ecx, PAGE_SIZE / LOGIC_SECT_SIZE ; 8 !
1815 <1> ; ; Note: Even if corresponding physical disk's sector
1816 <1> ; ; size different than 512 bytes, logical disk sector
1817 <1> ; ; size is 512 bytes and disk reading procedure
1818 <1> ; ; will be performed for reading 4096 bytes
1819 <1> ; ; (2*2048, 8*512).
1820 <1> ; ; ESI = Logical disk description table address
1821 <1> ; ; EBX = Memory page (buffer) address (physical!)
1822 <1> ; ; EAX = Sector address (offset address, logical sector number)
1823 <1> ; ; ECX = Sector count ; 8 sectors
1824 <1> ; push eax
1825 <1> ; call logical_disk_read
1826 <1> ; pop eax
1827 <1> ; jnc short swpin_read_ok
1828 <1> ;
1829 <1> ; mov eax, SWP_DISK_READ_ERR ; drive not ready or read error
1830 <1> ; mov [u.error], eax
1831 <1> ; jmp short swpin_retn
1832 <1> ;
1833 <1> ; swpin_read_ok:
1834 <1> ; ; EAX = Offset address (logical sector number)
1835 <1> ; call unlink_swap_block ; Deallocate swap block
1836 <1> ;
1837 <1> ; ; EBX = Memory page (buffer) address (physical!)
1838 <1> ; ; 20/07/2015
1839 <1> ; mov ebx, ebp ; virtual address (page fault address)
1840 <1> ; and bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
1841 <1> ; mov bl, [u.uno] ; current process number
1842 <1> ; ; EBX = Virtual (Linear) address & process number combination
1843 <1> ; call swap_queue_shift
1844 <1> ; ; eax = 0 ; 10/06/2016 (if ebx input > 0, eax output = 0)
1845 <1> ; sub eax, eax ; 0 ; Error Code = 0 (no error)
1846 <1> ; ; zf = 1
1847 <1> ; swpin_retn:

```



```

1848 <1> ; pop ecx
1849 <1> ; pop ebx
1850 <1> ; pop esi
1851 <1> ; retn
1852 <1> ;
1853 <1> ;swpin_dnp_err:
1854 <1> ; mov eax, SWP_DISK_NOT_PRESENT_ERR
1855 <1> ;swpin_err_retn:
1856 <1> ; mov [u.error], eax
1857 <1> ; stc
1858 <1> ; retn
1859 <1> ;
1860 <1> ;swpin_snp_err:
1861 <1> ; mov eax, SWP_SECTOR_NOT_PRESENT_ERR
1862 <1> ; jmp short swpin_err_retn
1863 <1> ;
1864 <1> ; 17/04/2021
1865 <1> ; ('swap_out' procedure call is disabled as temporary)
1866 <1> ;
1867 <1> swap_out:
1868 <1> ; 10/06/2016
1869 <1> ; 07/06/2016
1870 <1> ; 23/05/2016
1871 <1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
1872 <1> ; 24/10/2014 - 31/08/2015 (Retro UNIX 386 v1)
1873 <1> ;
1874 <1> ; INPUT ->
1875 <1> ; none
1876 <1> ;
1877 <1> ; OUTPUT ->
1878 <1> ; EAX = Physical page address (which is swapped out
1879 <1> ; for allocating a new page)
1880 <1> ; CF = 1 -> swap disk writing error (disk/file not present
1881 <1> ; or sector not present or drive not ready)
1882 <1> ; EAX = Error code
1883 <1> ; [u.error] = EAX
1884 <1> ; = The last error code for the process
1885 <1> ; (will be reset after returning to user)
1886 <1> ;
1887 <1> ; Modified Registers -> none (except EAX)
1888 <1> ;
1889 <1> ;
1890 <1> ; cmp word [swpq_count], 1
1891 <1> ; jc swpout_im_err ; 'insufficient memory'
1892 <1> ;
1893 <1> ; cmp dword [swp_drv], 1
1894 <1> ; jc short swpout_dnp_err ; 'swap disk/file not present'
1895 <1> ;
1896 <1> ; cmp dword [swpd_free], 1
1897 <1> ; jc swpout_nfspc_err ; 'no free space on swap disk'
1898 <1> ;
1899 <1> ; push ebx ; *
1900 <1> ;swpout_1:
1901 <1> ; 10/06/2016
1902 <1> ; xor ebx, ebx ; shift the queue and return a PTE value
1903 <1> ; call swap_queue_shift
1904 <1> ; and eax, eax ; 0 = empty queue (improper entries)
1905 <1> ; jz swpout_npts_err ; There is not any proper PTE
1906 <1> ; ; pointer in the swap queue
1907 <1> ;
1908 <1> ; EAX = PTE value of the page
1909 <1> ; EBX = PTE address of the page
1910 <1> ; and ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
1911 <1> ;
1912 <1> ; 07/06/2016
1913 <1> ; 19/05/2016
1914 <1> ; ; check this page is in timer events or not
1915 <1> ;swpout_timer_page_0:
1916 <1> ; push edx ; **
1917 <1> ;
1918 <1> ; 07/06/2016
1919 <1> ; cmp byte [timer_events], 0
1920 <1> ; jna short swpout_2
1921 <1> ;
1922 <1> ; mov dl, [timer_events]
1923 <1> ;
1924 <1> ; push ecx ; ***
1925 <1> ; push ebx ; ****
1926 <1> ; mov ebx, timer_set ; beginning address of timer event
1927 <1> ; ; structures
1928 <1> ;swpout_timer_page_1:
1929 <1> ; mov cl, [ebx]
1930 <1> ; or cl, cl ; 0 = free, >0 = process number
1931 <1> ; jz short swpout_timer_page_3
1932 <1> ; mov ecx, [ebx+12] ; response (signal return) address
1933 <1> ; and cx, PTE_A_CLEAR ; clear offset part (right 12 bits)
1934 <1> ; ; of the response byte address, to
1935 <1> ; ; get beginning of the page address)
1936 <1> ; cmp eax, ecx
1937 <1> ; jne short swpout_timer_page_2 ; not same page
1938 <1> ;
1939 <1> ; ; !same page!
1940 <1> ;
1941 <1> ; ; NOTE: // 19/05/2016 // - TRDOS 386 feature only ! -
1942 <1> ; ; This page will be used by the kernel to put timer event
1943 <1> ; ; response (signal return) byte at the requested address;
1944 <1> ; ; in order to prevent a possible wrong write (while
1945 <1> ; ; this page is swapped out) on physical memory,
1946 <1> ; ; we must protect this page against to be swapped out!
1947 <1> ;
1948 <1> ; pop ebx ; ****
1949 <1> ; pop ecx ; ***
1950 <1> ; pop edx ; **
1951 <1> ; jmp short swpout_1 ; do not swap out this page !
1952 <1> ;
1953 <1> ;swpout_timer_page_2:
1954 <1> ; 07/06/2016
1955 <1> ; dec dl
1956 <1> ; jz short swpout_timer_page_4
1957 <1> ;swpout_timer_page_3:
1958 <1> ; cmp ebx, timer_set + 240 ; last timer event (15*16)
1959 <1> ; jnb short swpout_timer_page_4
1960 <1> ; add ebx, 16
1961 <1> ; jmp short swpout_timer_page_1
1962 <1> ;
1963 <1> ;swpout_timer_page_4:
1964 <1> ; pop ebx ; ****
1965 <1> ; pop ecx ; ***
1966 <1> ;swpout_2:
1967 <1> ; mov edx, ebx ; Page table entry address
1968 <1> ; mov ebx, eax ; Buffer (Page) Address
1969 <1> ;
1970 <1> ; call link_swap_block
1971 <1> ; jnc short swpout_3 ; It may not be needed here

```

```

1972 <1> ; ; because [swpd_free] value
1973 <1> ; ; was checked at the beginging.
1974 <1> ; pop edx ; **
1975 <1> ; pop ebx ; *
1976 <1> ; jmp short swpout_nfspc_err
1977 <1> ; swpout_3:
1978 <1> ; test eax, 80000000h ; test bit 31 (this may not be needed!)
1979 <1> ; jnz short swpout_nfspc_err ; 10/06/2016 (bit 31 = 1 !)
1980 <1> ; ;
1981 <1> ; push esi ; **
1982 <1> ; push ecx ; ***
1983 <1> ; push eax ; sector address ; (31 bit !, bit 31 = 0)
1984 <1> ; mov esi, [swp_drv]
1985 <1> ; mov ecx, PAGE_SIZE / LOGIC_SECT_SIZE ; 8 !
1986 <1> ; ; Note: Even if corresponding physical disk's sector
1987 <1> ; ; size different than 512 bytes, logical disk sector
1988 <1> ; ; size is 512 bytes and disk writing procedure
1989 <1> ; ; will be performed for writing 4096 bytes
1990 <1> ; ; (2*2048, 8*512).
1991 <1> ; ; ESI = Logical disk description table address
1992 <1> ; ; EBX = Buffer (Page) address
1993 <1> ; ; EAX = Sector address (Offset address, logical sector number)
1994 <1> ; ; ECX = Sector count ; 8 sectors
1995 <1> ; ; edx = PTE address
1996 <1> ; call logical_disk_write
1997 <1> ; ; edx = PTE address
1998 <1> ; pop ecx ; sector address
1999 <1> ; jnc short swpout_write_ok
2000 <1> ; ;
2001 <1> ; ; call unlink_swap_block ; this block must be left as 'in use'
2002 <1> ; swpout_dw_err:
2003 <1> ; mov eax, SWP_DISK_WRITE_ERR ; drive not ready or write error
2004 <1> ; mov [u.error], eax
2005 <1> ; jmp short swpout_retn
2006 <1> ; ;
2007 <1> ; swpout_write_ok:
2008 <1> ; ; EBX = Buffer (page) address
2009 <1> ; ; EDX = Page Table Entry address
2010 <1> ; ; ECX = Swap disk sector (file block) address (31 bit)
2011 <1> ; shl ecx, 1 ; 31 bit sector address from bit 1 to bit 31
2012 <1> ; mov [edx], ecx
2013 <1> ; ; bit 0 = 0 (swapped page)
2014 <1> ; mov eax, ebx
2015 <1> ; swpout_retn:
2016 <1> ; pop ecx ; ***
2017 <1> ; pop esi ; **
2018 <1> ; pop ebx ; *
2019 <1> ; ret
2020 <1> ; ;
2021 <1> ; swpout_dnp_err:
2022 <1> ; mov eax, SWP_DISK_NOT_PRESENT_ERR ; disk not present
2023 <1> ; jmp short swpout_err_retn
2024 <1> ; swpout_nfspc_err:
2025 <1> ; mov eax, SWP_NO_FREE_SPACE_ERR ; no free space
2026 <1> ; swpout_err_retn:
2027 <1> ; mov [u.error], eax
2028 <1> ; stc
2029 <1> ; ret
2030 <1> ; swpout_npts_err:
2031 <1> ; mov eax, SWP_NO_PAGE_TO_SWAP_ERR
2032 <1> ; pop ebx
2033 <1> ; jmp short swpout_err_retn
2034 <1> ; swpout_im_err:
2035 <1> ; mov eax, ERR_MINOR_IM ; insufficient (out of) memory
2036 <1> ; jmp short swpout_err_retn
2037 <1> ; ;
2038 <1> ; ; 17/04/2021
2039 <1> ; ; ('swap_queue_shift' procedure call is disabled as temporary)
2040 <1> ; ;
2041 <1> ; swap_queue_shift:
2042 <1> ; ; 26/03/2017
2043 <1> ; ; 10/06/2016
2044 <1> ; ; 09/06/2016 - TRDOS 386 (TRDOS v2.0)
2045 <1> ; ; 23/10/2014 - 20/07/2015 (Retro UNIX 386 v1)
2046 <1> ; ;
2047 <1> ; ; INPUT ->
2048 <1> ; ; EBX = Virtual (linear) address (bit 12 to 31)
2049 <1> ; ; and process number combination (bit 0 to 11)
2050 <1> ; ; EBX = 0 -> shift/drop from the head (offset 0)
2051 <1> ; ;
2052 <1> ; ; OUTPUT ->
2053 <1> ; ; If EBX input > 0
2054 <1> ; ; the queue will be shifted 4 bytes (dword),
2055 <1> ; ; from the tail to the head, up to entry offset
2056 <1> ; ; which points to EBX input value or nothing
2057 <1> ; ; to do if EBX value is not found on the queue.
2058 <1> ; ; (The entry -with EBX value- will be removed
2059 <1> ; ; from the queue if it is found.)
2060 <1> ; ;
2061 <1> ; ; EAX = 0
2062 <1> ; ;
2063 <1> ; ; If EBX input = 0
2064 <1> ; ; the queue will be shifted 4 bytes (dword),
2065 <1> ; ; from the tail to the head, if the PTE address
2066 <1> ; ; which is pointed in head of the queue is marked
2067 <1> ; ; as "accessed" or it is marked as "non present".
2068 <1> ; ; (If "accessed" flag of the PTE -which is pointed
2069 <1> ; ; in the head- is set -to 1-, it will be reset
2070 <1> ; ; -to 0- and then, the queue will be rotated
2071 <1> ; ; -without dropping pointer of the PTE from
2072 <1> ; ; the queue- for 4 bytes on head to tail direction.
2073 <1> ; ; Pointer in the head will be moved into the tail,
2074 <1> ; ; other PTEs will be shifted on head direction.)
2075 <1> ; ;
2076 <1> ; ; Swap queue will be shifted up to the first
2077 <1> ; ; 'present' or 'non accessed' page will be found
2078 <1> ; ; (as pointed) on the queue head (then it will be
2079 <1> ; ; removed/dropped from the queue).
2080 <1> ; ;
2081 <1> ; ; EAX (> 0) = PTE value of the page which is
2082 <1> ; ; (it's pointer -virtual address-) dropped
2083 <1> ; ; (removed) from swap queue.
2084 <1> ; ; EBX = PTE address of the page (if EAX > 0)
2085 <1> ; ; which is (it's pointer -virtual address-)
2086 <1> ; ; dropped (removed) from swap queue.
2087 <1> ; ;
2088 <1> ; ; EAX = 0 -> empty swap queue !
2089 <1> ; ;
2090 <1> ; ; Modified Registers -> EAX, EBX
2091 <1> ; ;
2092 <1> ; movzx eax, word [swpq_count] ; Max. 1024
2093 <1> ; and ax, ax
2094 <1> ; jz short swpq_retn
2095 <1> ; push edi

```

```

2096 <1> ; push esi
2097 <1> ; push ecx
2098 <1> ; mov esi, swap_queue
2099 <1> ; mov ecx, eax
2100 <1> ; or ebx, ebx
2101 <1> ; jz short swpqs_7
2102 <1> ; swpqs_1:
2103 <1> ; lodsd
2104 <1> ; cmp eax, ebx
2105 <1> ; je short swpqs_2
2106 <1> ; loop swpqs_1
2107 <1> ; ; 10/06/2016
2108 <1> ; sub eax, eax
2109 <1> ; jmp short swpqs_6
2110 <1> ; swpqs_2:
2111 <1> ; mov edi, esi
2112 <1> ; sub edi, 4
2113 <1> ; swpqs_3:
2114 <1> ; dec word [swpq_count]
2115 <1> ; jz short swpqs_5
2116 <1> ; swpqs_4:
2117 <1> ; dec ecx
2118 <1> ; rep movsd ; shift up (to the head)
2119 <1> ; swpqs_5:
2120 <1> ; xor eax, eax
2121 <1> ; mov [edi], eax
2122 <1> ; swpqs_6:
2123 <1> ; pop ecx
2124 <1> ; pop esi
2125 <1> ; pop edi
2126 <1> ; swpqs_retn:
2127 <1> ; ret
2128 <1> ; swpqs_7:
2129 <1> ; mov edi, esi ; head
2130 <1> ; lodsd
2131 <1> ; ; 20/07/2015
2132 <1> ; mov ebx, eax
2133 <1> ; and ebx, ~PAGE_OFF ; ~0FFFh
2134 <1> ; ; ebx = virtual address (at page boundary)
2135 <1> ; and eax, PAGE_OFF ; 0FFFh
2136 <1> ; ; ax = process number (1 to 4095)
2137 <1> ; cmp al, [u.uno]
2138 <1> ; ; Max. 16 (nproc) processes for Retro UNIX 386 v1
2139 <1> ; jne short swpqs_8
2140 <1> ; mov eax, [u.pgdir]
2141 <1> ; jmp short swpqs_9
2142 <1> ; swpqs_8:
2143 <1> ; ; 09/06/2016
2144 <1> ; cmp byte [eax+p.stat-1], 0
2145 <1> ; jna short swpqs_3 ; free (or terminated) process
2146 <1> ; cmp byte [eax+p.stat-1], 2 ; waiting
2147 <1> ; ja short swpqs_3 ; zombie (3) or undefined ?
2148 <1> ;
2149 <1> ; shl ax, 2
2150 <1> ; shl al, 2
2151 <1> ; mov eax, [eax+p.upage-4]
2152 <1> ; or eax, eax
2153 <1> ; jz short swpqs_3 ; invalid upage
2154 <1> ; add eax, u.pgdir - user
2155 <1> ; ; u.pgdir value for the process
2156 <1> ; ; is in [eax]
2157 <1> ; mov eax, [eax]
2158 <1> ; and eax, eax
2159 <1> ; jz short swpqs_3 ; invalid page directory
2160 <1> ; swpqs_9:
2161 <1> ; push edx
2162 <1> ; ; eax = page directory
2163 <1> ; ; ebx = virtual address
2164 <1> ; call get_pte
2165 <1> ; mov ebx, edx ; PTE address
2166 <1> ; pop edx
2167 <1> ; ; 10/06/2016
2168 <1> ; jc short swpqs_13 ; empty PDE
2169 <1> ; ; EAX = PTE value
2170 <1> ; test al, PTE_A_PRESENT ; bit 0 = 1
2171 <1> ; jz short swpqs_13 ; Drop non-present page
2172 <1> ; ; from the queue (head)
2173 <1> ; test al, PTE_A_WRITE ; bit 1 = 0 (read only)
2174 <1> ; jz short swpqs_13 ; Drop read only page
2175 <1> ; ; from the queue (head)
2176 <1> ; test al, PTE_A_ACCESS ; bit 5 = 1 (Accessed)
2177 <1> ; jnz short swpqs_11 ; present
2178 <1> ; ; accessed page
2179 <1> ; btr eax, PTE_A_ACCESS_BIT ; reset 'accessed' bit
2180 <1> ; jc short swpqs_11 ; accessed page
2181 <1> ;
2182 <1> ; dec ecx
2183 <1> ; mov [swpq_count], cx
2184 <1> ; jz short swpqs_10
2185 <1> ; ; esi = head + 4
2186 <1> ; ; edi = head
2187 <1> ; rep movsd ; n = 1 to k-1, [n - 1] = [n]
2188 <1> ; swpqs_10:
2189 <1> ; mov [edi], ecx ; 0
2190 <1> ; jmp short swpqs_6 ; 26/03/2017
2191 <1> ;
2192 <1> ; swpqs_11:
2193 <1> ; mov [ebx], eax ; save changed attribute
2194 <1> ; ; Rotation (head -> tail)
2195 <1> ; dec ecx ; entry count -> last entry number
2196 <1> ; jz short swpqs_10
2197 <1> ; ; esi = head + 4
2198 <1> ; ; edi = head
2199 <1> ; mov eax, [edi] ; 20/07/2015
2200 <1> ; rep movsd ; n = 1 to k-1, [n - 1] = [n]
2201 <1> ; mov [edi], eax ; head -> tail ; [k] = [1]
2202 <1> ;
2203 <1> ; mov cx, [swpq_count]
2204 <1> ;
2205 <1> ; swpqs_12:
2206 <1> ; mov esi, swap_queue ; head
2207 <1> ; jmp swpqs_7
2208 <1> ;
2209 <1> ; swpqs_13:
2210 <1> ; dec ecx
2211 <1> ; mov [swpq_count], cx
2212 <1> ; jz swpqs_5
2213 <1> ; jmp short swpqs_12
2214 <1> ;
2215 <1> ; ; 17/04/2021
2216 <1> ; ; ('add_to_swap_queue' procedure call is disabled as temporary)
2217 <1> ;
2218 <1> ; add_to_swap_queue:
2219 <1> ; ; 20/02/2017

```

```

2220 <1> ; 20/07/2015
2221 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
2222 <1> ;
2223 <1> ; Adds new page to swap queue
2224 <1> ; (page directories and page tables must not be added
2225 <1> ; to swap queue)
2226 <1> ;
2227 <1> ; INPUT ->
2228 <1> ; EBX = Linear (virtual) addr for current process
2229 <1> ; [u.uno]
2230 <1> ; 20/02/2017
2231 <1> ; (Linear address = CORE + user's virtual address)
2232 <1> ;
2233 <1> ; OUTPUT ->
2234 <1> ; EAX = [swpq_count]
2235 <1> ; (after the PTE has been added)
2236 <1> ; EAX = 0 -> Swap queue is full, (1024 entries)
2237 <1> ; the PTE could not be added.
2238 <1> ;
2239 <1> ; Modified Registers -> EAX
2240 <1> ;
2241 <1> ; push ebx
2242 <1> ; and bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
2243 <1> ; mov bl, [u.uno] ; current process number
2244 <1> ; call swap_queue_shift ; drop from the queue if
2245 <1> ; ; it is already on the queue
2246 <1> ; ; then add it to the tail of the queue
2247 <1> ; movzx eax, word [swpq_count]
2248 <1> ; cmp ax, 1024
2249 <1> ; jb short atsq_1
2250 <1> ; sub ax, ax
2251 <1> ; pop ebx
2252 <1> ; retn
2253 <1> ; atsq_1:
2254 <1> ; push esi
2255 <1> ; mov esi, swap_queue
2256 <1> ; and ax, ax
2257 <1> ; jz short atsq_2
2258 <1> ; shl ax, 2 ; convert to offset
2259 <1> ; add esi, eax
2260 <1> ; shr ax, 2
2261 <1> ; atsq_2:
2262 <1> ; inc ax
2263 <1> ; mov [esi], ebx ; Virtual address + [u.uno] combination
2264 <1> ; mov [swpq_count], ax
2265 <1> ; pop esi
2266 <1> ; pop ebx
2267 <1> ; retn
2268 <1> ;
2269 <1> ; 17/04/2021
2270 <1> ; ('unlink_swap_block' procedure call is disabled as temporary)
2271 <1> ;
2272 <1> ; unlink_swap_block:
2273 <1> ; 15/09/2015
2274 <1> ; 30/04/2015
2275 <1> ; 18/04/2015
2276 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
2277 <1> ;
2278 <1> ; INPUT ->
2279 <1> ; EAX = swap disk/file offset address
2280 <1> ; (bit 1 to bit 31)
2281 <1> ; OUTPUT ->
2282 <1> ; [swpd_free] is increased
2283 <1> ; (corresponding SWAP DISK ALLOC. TABLE bit is SET)
2284 <1> ;
2285 <1> ; Modified Registers -> EAX
2286 <1> ;
2287 <1> ; push ebx
2288 <1> ; push edx
2289 <1> ;
2290 <1> ; shr eax, SECTOR_SHIFT+1 ;3+1 ; shift sector address to
2291 <1> ; ; 3 bits right
2292 <1> ; ; to get swap block/page number
2293 <1> ; mov edx, eax
2294 <1> ; 15/09/2015
2295 <1> ; shr edx, 3 ; to get offset to S.A.T.
2296 <1> ; ; (1 allocation bit = 1 page)
2297 <1> ; ; (1 allocation bytes = 8 pages)
2298 <1> ; and dl, 0Fch ; clear lower 2 bits
2299 <1> ; ; (to get 32 bit position)
2300 <1> ;
2301 <1> ; mov ebx, swap_alloc_table ; Swap Allocation Table address
2302 <1> ; add ebx, edx
2303 <1> ; and eax, 1Fh ; lower 5 bits only
2304 <1> ; ; (allocation bit position)
2305 <1> ; cmp eax, [swpd_next] ; is the new free block addr. lower
2306 <1> ; ; than the address in 'swpd_next' ?
2307 <1> ; ; (next/first free block value)
2308 <1> ; jnb short uswpbl_1 ; no
2309 <1> ; mov [swpd_next], eax ; yes
2310 <1> ; uswpbl_1:
2311 <1> ; bts [ebx], eax ; unlink/release/deallocate block
2312 <1> ; ; set relevant bit to 1.
2313 <1> ; ; set CF to the previous bit value
2314 <1> ; cmc ; complement carry flag
2315 <1> ; jc short uswpbl_2 ; do not increase swfd_free count
2316 <1> ; ; if the block is already deallocated
2317 <1> ; ; before.
2318 <1> ; inc dword [swpd_free]
2319 <1> ; uswpbl_2:
2320 <1> ; pop edx
2321 <1> ; pop ebx
2322 <1> ; retn
2323 <1> ;
2324 <1> ; 17/04/2021
2325 <1> ; ('link_swap_block' procedure call is disabled as temporary)
2326 <1> ;
2327 <1> ; link_swap_block:
2328 <1> ; 01/07/2015
2329 <1> ; 18/04/2015
2330 <1> ; 24/10/2014 (Retro UNIX 386 v1 - beginning)
2331 <1> ;
2332 <1> ; INPUT -> none
2333 <1> ;
2334 <1> ; OUTPUT ->
2335 <1> ; EAX = OFFSET ADDRESS OF THE ALLOCATED BLOCK (4096 bytes)
2336 <1> ; in sectors (corresponding
2337 <1> ; SWAP DISK ALLOCATION TABLE bit is RESET)
2338 <1> ;
2339 <1> ; CF = 1 and EAX = 0
2340 <1> ; ; if there is not a free block to be allocated
2341 <1> ;
2342 <1> ; Modified Registers -> none (except EAX)
2343 <1> ;

```

```

2344 <1>
2345 <1> ; mov    eax, [swpd_free]
2346 <1> ; and    eax, eax
2347 <1> ; jz     short out_of_swpspc
2348 <1> ;
2349 <1> ; push   ebx
2350 <1> ; push   ecx
2351 <1> ;
2352 <1> ; mov    ebx, swap_alloc_table ; Swap Allocation Table offset
2353 <1> ; mov    ecx, ebx
2354 <1> ; add    ebx, [swpd_next] ; Free block searching starts from here
2355 <1> ; ; next_free_swap_block >> 5
2356 <1> ; add    ecx, [swpd_last] ; Free block searching ends here
2357 <1> ; ; (total_swap_blocks - 1) >> 5
2358 <1> ; lswbl_scan:
2359 <1> ; cmp    ebx, ecx
2360 <1> ; ja     short lswbl_notfound
2361 <1> ;
2362 <1> ; bsf    eax, [ebx] ; Scans source operand for first bit set (1).
2363 <1> ; ; Clears ZF if a bit is found set (1) and
2364 <1> ; ; loads the destination with an index to
2365 <1> ; ; first set bit. (0 -> 31)
2366 <1> ; ; Sets ZF to 1 if no bits are found set.
2367 <1> ; ; 01/07/2015
2368 <1> ; jnz    short lswbl_found ; ZF = 0 -> a free block has been found
2369 <1> ;
2370 <1> ; ; NOTE: a Swap Disk Allocation Table bit
2371 <1> ; ; with value of 1 means
2372 <1> ; ; the corresponding page is free
2373 <1> ; ; (Retro UNIX 386 v1 feaure only!)
2374 <1> ; add    ebx, 4
2375 <1> ; ; We return back for searching next page block
2376 <1> ; ; NOTE: [swpd_free] is not ZERO; so,
2377 <1> ; ; we always will find at least 1 free block here.
2378 <1> ; jmp    short lswbl_scan
2379 <1> ;
2380 <1> ; lswbl_notfound:
2381 <1> ; sub    ecx, swap_alloc_table
2382 <1> ; mov    [swpd_next], ecx ; next/first free page = last page
2383 <1> ; ; (unlink_swap_block procedure will change it)
2384 <1> ; xor    eax, eax
2385 <1> ; mov    [swpd_free], eax
2386 <1> ; stc
2387 <1> ; lswbl_ok:
2388 <1> ; pop    ecx
2389 <1> ; pop    ebx
2390 <1> ; retn
2391 <1> ;
2392 <1> ; out_of_swpspc:
2393 <1> ; stc
2394 <1> ; retn
2395 <1> ;
2396 <1> ; lswbl_found:
2397 <1> ; mov    ecx, ebx
2398 <1> ; sub    ecx, swap_alloc_table
2399 <1> ; mov    [swpd_next], ecx ; Set first free block searching start
2400 <1> ; ; address/offset (to the next)
2401 <1> ; dec    dword [swpd_free] ; 1 block has been allocated (X = X-1)
2402 <1> ;
2403 <1> ; btr    [ebx], eax ; The destination bit indexed by the source value
2404 <1> ; ; is copied into the Carry Flag and then cleared
2405 <1> ; ; in the destination.
2406 <1> ;
2407 <1> ; ; Reset the bit which is corresponding to the
2408 <1> ; ; (just) allocated block.
2409 <1> ; shl    ecx, 5 ; (block offset * 32) + block index
2410 <1> ; add    eax, ecx ; = block number
2411 <1> ; shl    eax, SECTOR_SHIFT ; 3, sector (offset) address of the block
2412 <1> ; ; 1 block = 8 sectors
2413 <1> ;
2414 <1> ; ; EAX = offset address of swap disk/file sector (beginning of the block)
2415 <1> ;
2416 <1> ; ; NOTE: The relevant page table entry will be updated
2417 <1> ; ; according to this EAX value...
2418 <1> ;
2419 <1> ; jmp    short lswbl_ok
2420 <1> ;
2421 <1> ; 17/04/2021
2422 <1> ; ('logical_disk_read' procedure call is disabled as temporary)
2423 <1> ;
2424 <1> ; logical_disk_read:
2425 <1> ; ; 20/07/2015
2426 <1> ; ; 09/03/2015 (temporary code here)
2427 <1> ;
2428 <1> ; INPUT ->
2429 <1> ; ; ESI = Logical disk description table address
2430 <1> ; ; EBX = Memory page (buffer) address (physical!)
2431 <1> ; ; EAX = Sector address (offset address, logical sector number)
2432 <1> ; ; ECX = Sector count
2433 <1> ;
2434 <1> ;
2435 <1> ; retn
2436 <1> ;
2437 <1> ; 17/04/2021
2438 <1> ; ('logical_disk_write' procedure call is disabled as temporary)
2439 <1> ;
2440 <1> ; logical_disk_write:
2441 <1> ; ; 20/07/2015
2442 <1> ; ; 09/03/2015 (temporary code here)
2443 <1> ;
2444 <1> ; INPUT ->
2445 <1> ; ; ESI = Logical disk description table address
2446 <1> ; ; EBX = Memory page (buffer) address (physical!)
2447 <1> ; ; EAX = Sector address (offset address, logical sector number)
2448 <1> ; ; ECX = Sector count
2449 <1> ;
2450 <1> ; retn
2451 <1> ;
2452 <1> ; get_physical_addr:
2453 <1> ; ; 17/04/2021 - TRDOS 386 v2.0.4
2454 <1> ; ; (temporary modifications)
2455 <1> ;
2456 <1> ; ; 26/03/2017
2457 <1> ; ; 20/02/2017
2458 <1> ; ; 27/05/2016 - TRDOS 386 (TRDOS v2.0)
2459 <1> ; ; 18/10/2015
2460 <1> ; ; 29/07/2015
2461 <1> ; ; 20/07/2015
2462 <1> ; ; 04/06/2015
2463 <1> ; ; 20/05/2015
2464 <1> ; ; 28/04/2015
2465 <1> ; ; 18/04/2015
2466 <1> ; ; Get physical address
2467 <1> ; ; (allocates a new page for user if it is not present)

```

```

2468      ;
2469      ; (This subroutine is needed for mapping user's virtual
2470      ; (buffer) address to physical address (of the buffer).)
2471      ; ('sys write', 'sys read' system calls...)
2472      ;
2473      ; INPUT ->
2474      ; EBX = virtual address
2475      ; u.pgdir = page directory (physical) address
2476      ;
2477      ; OUTPUT ->
2478      ; EAX = physical address
2479      ; EBX = linear address
2480      ; EDX = physical address of the page frame
2481      ;         (with attribute bits)
2482      ; ECX = byte count within the page frame
2483      ;
2484      ; Modified Registers -> EAX, EBX, ECX, EDX
2485      ;
2486      00005C04 81C300004000      add     ebx, CORE ; 18/10/2015
2487      get_physical_addr_x: ; 27/05/2016
2488      00005C0A A1[AC8E0100]      mov     eax, [u.pgdir]
2489      00005C0F E824FDFFFF      call    get_pte
2490      ; EDX = Page table entry address (if CF=0)
2491      ;         Page directory entry address (if CF=1)
2492      ;         (Bit 0 value is 0 if PT is not present)
2493      ; EAX = Page table entry value (page address)
2494      ;         CF = 1 -> PDE not present or invalid ?
2495      00005C14 731C      jnc     short gpa_1
2496      ;
2497      00005C16 E80BFCFFFF      call    allocate_page
2498      00005C1B 723F      jc      short gpa_im_err ; 'insufficient memory' error
2499      gpa_0:
2500      00005C1D E875FCFFFF      call    clear_page
2501      ; EAX = Physical (base) address of the allocated (new) page
2502      00005C22 0C07      or      al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER ; 4+2+1 = 7
2503      ; lower 3 bits are used as U/S, R/W, P flags
2504      ;         (user, writable, present page)
2505      00005C24 8902      mov     [edx], eax ; Let's put the new page directory entry here !
2506      00005C26 A1[AC8E0100]      mov     eax, [u.pgdir]
2507      00005C2B E808FDFFFF      call    get_pte
2508      00005C30 722A      jc      short gpa_im_err ; 'insufficient memory' error
2509      gpa_1:
2510      ; EAX = PTE value, EDX = PTE address
2511      00005C32 A801      test    al, PTE_A_PRESENT
2512      00005C34 7516      jnz     short gpa_3 ; 26/03/2017
2513      00005C36 09C0      or      eax, eax
2514      00005C38 7446      jz      short gpa_7 ; Allocate a new page
2515      ;
2516      ; 17/04/2021 (TRDOS 386 v2.0.4)
2517      ; ('reload_page' procedure call is disabled as temporary)
2518      00005C3A EB20      jmp     short gpa_im_err ; temporary !
2519      ;
2520      ; 20/07/2015
2521      ; push     ebp
2522      ; mov      ebp, ebx ; virtual (linear) address
2523      ; reload swapped page
2524      ; call     reload_page ; 28/04/2015
2525      ; pop      ebp
2526      ; jc      short gpa_retn
2527      gpa_2:
2528      ; 26/03/2017
2529      ; 20/02/2017
2530      ; If a page will contain a Signal Response Byte
2531      ; it must not be swapped out, because
2532      ; timer service or irq callback service
2533      ; will write a signal return/response byte
2534      ; directly by using physical address of Signal
2535      ; Response Byte. (Even if process is not running,
2536      ; or it is running with swapped out pages.)
2537      ;
2538      ; 'no_page_swap' will be set by 'systemtimer' or
2539      ; 'syscalbac' sistem functions/calls. (*)
2540      ;
2541      00005C3C 803D[3E880100]00      cmp     byte [no_page_swap], 0
2542      00005C43 761D      jna     short gpa_4 ; this page can be swapped out
2543      ; this page must not be swapped out
2544      ; but 'no_page_swap' must be reset here
2545      ; immediately for other callers (*)
2546      ; (otherwise, swap queue would not be long enough)
2547      00005C45 E844000000      call    gpa_8 ; 26/03/2017
2548      00005C4A EB16      jmp     short gpa_5
2549      gpa_3:
2550      ; 26/03/2017
2551      00005C4C 803D[3E880100]00      cmp     byte [no_page_swap], 0
2552      00005C53 7611      jna     short gpa_6 ; this page can be swapped out
2553      00005C55 E834000000      call    gpa_8
2554      00005C5A EB0A      jmp     short gpa_6
2555      ;
2556      gpa_im_err:
2557      00005C5C B804000000      mov     eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
2558      ; Major error = 0 (No protection fault)
2559      00005C61 C3      retn
2560      gpa_4:
2561      ; 17/04/2021 (TRDOS 386 v2.0.4)
2562      ; ('add_to_swap_queue' procedure call is disabled as temporary)
2563      ;
2564      ; 20/07/2015
2565      ; 20/05/2015
2566      ; add this page to swap queue
2567      ; push     eax
2568      ; EBX = Linear (CORE+virtual) address ; 20/02/2017
2569      ; call     add_to_swap_queue
2570      ; pop      eax
2571      gpa_5:
2572      ; PTE address in EDX
2573      ; virtual address in EBX
2574      ; EAX = memory page address
2575      00005C62 0C07      or      al, PTE_A_PRESENT + PTE_A_USER + PTE_A_WRITE
2576      ; present flag, bit 0 = 1
2577      ; user flag, bit 2 = 1
2578      ; writable flag, bit 1 = 1
2579      00005C64 8902      mov     [edx], eax ; update PTE value
2580      gpa_6:
2581      ; 18/10/2015
2582      00005C66 89D9      mov     ecx, ebx
2583      00005C68 81E1FF0F0000      and     ecx, PAGE_OFF
2584      00005C6E 89C2      mov     edx, eax
2585      00005C70 662500F0      and     ax, PTE_A_CLEAR
2586      00005C74 01C8      add     eax, ecx
2587      00005C76 F7D9      neg     ecx ; 1 -> -1 (0FFFFFFFh), 4095 (0FFFh) -> -4095
2588      00005C78 81C100100000      add     ecx, PAGE_SIZE
2589      00005C7E F8      cld
2590      gpa_retn:
2591      00005C7F C3      retn

```

```

2592 <1> gpa_7:
2593 00005C80 E8A1FBFFFF <1> call allocate_page
2594 00005C85 72D5 <1> jc short gpa_im_err ; 'insufficient memory' error
2595 00005C87 E80BFCFFFF <1> call clear_page
2596 00005C8C EBAE <1> jmp short gpa_2
2597 <1>
2598 <1> gpa_8: ; 26/03/2017
2599 00005C8E C605[3E880100]00 <1> mov byte [no_page_swap], 0
2600 <1>
2601 00005C95 C3 <1> retn ; 17/04/2021 (temporary)
2602 <1>
2603 <1> ; 17/04/2021 (TRDOS 386 v2.0.4)
2604 <1> ; ('swap_queue_shift' procedure call is disabled as temporary)
2605 <1>
2606 <1> ;
2607 <1> ; push ebx
2608 <1> ; push eax ; 26/03/2017
2609 <1> ; and bx, ~PAGE_OFF ; ~0FFFh ; reset bits, 0 to 11
2610 <1> ; mov bl, [u.uno] ; current process number
2611 <1> ; call swap_queue_shift ; drop from the queue if
2612 <1> ; ; it is already on the queue
2613 <1> ; pop eax ; 26/03/2017
2614 <1> ; pop ebx
2615 <1> ;
2616 <1> ; retn
2617 <1> ; 17/04/2021
2618 <1> ; ('reload_page' procedure call is disabled as temporary)
2619 <1>
2620 <1> reload_page:
2621 <1> ; 20/07/2015
2622 <1> ; 28/04/2015 (Retro UNIX 386 v1 - beginning)
2623 <1>
2624 <1> ; Reload (Restore) swapped page at memory
2625 <1>
2626 <1> ; INPUT ->
2627 <1> ; EBP = virtual (linear) memory address
2628 <1> ; EAX = PTE value (swap disk sector address)
2629 <1> ; (Swap disk sector address = bit 1 to bit 31 of EAX)
2630 <1> ; OUTPUT ->
2631 <1> ; EAX = PHYSICAL (real/flat) ADDRESS OF RELOADED PAGE
2632 <1> ;
2633 <1> ; CF = 1 and EAX = error code
2634 <1> ;
2635 <1> ; Modified Registers -> none (except EAX)
2636 <1> ;
2637 <1> ; shr eax, 1 ; Convert PTE value to swap disk address
2638 <1> ; push ebx ;
2639 <1> ; mov ebx, eax ; Swap disk (offset) address
2640 <1> ; call allocate_page
2641 <1> ; jc short rlp_im_err
2642 <1> ; xchg eax, ebx
2643 <1> ; ; EBX = Physical memory (page) address
2644 <1> ; ; EAX = Swap disk (offset) address
2645 <1> ; ; EBP = Virtual (linear) memory address
2646 <1> ; call swap_in
2647 <1> ; jc short rlp_swp_err ; (swap disk/file read error)
2648 <1> ; mov eax, ebx
2649 <1> ; rlp_retn:
2650 <1> ; pop ebx
2651 <1> ; retn
2652 <1>
2653 <1> ; rlp_im_err:
2654 <1> ; mov eax, ERR_MINOR_IM ; Insufficient memory (minor) error!
2655 <1> ; ; Major error = 0 (No protection fault)
2656 <1> ; jmp short rlp_retn
2657 <1>
2658 <1> ; rlp_swp_err:
2659 <1> ; mov eax, SWP_DISK_READ_ERR ; Swap disk read error !
2660 <1> ; jmp short rlp_retn
2661 <1>
2662 <1> copy_page_dir:
2663 <1> ; 17/04/2021 (temporary modifications)
2664 <1> ; 19/09/2015
2665 <1> ; temporary - 07/09/2015
2666 <1> ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
2667 <1> ;
2668 <1> ; INPUT ->
2669 <1> ; [u.pgdir] = PHYSICAL (real/flat) ADDRESS of the parent's
2670 <1> ; page directory.
2671 <1> ; OUTPUT ->
2672 <1> ; EAX = PHYSICAL (real/flat) ADDRESS of the child's
2673 <1> ; page directory.
2674 <1> ; (New page directory with new page table entries.)
2675 <1> ; (New page tables with read only copies of the parent's
2676 <1> ; pages.)
2677 <1> ; EAX = 0 -> Error (CF = 1)
2678 <1> ;
2679 <1> ; Modified Registers -> none (except EAX)
2680 <1> ;
2681 00005C96 E88BFBFFFF <1> call allocate_page
2682 00005C9B 723E <1> jc short cpd_err
2683 <1> ;
2684 00005C9D 55 <1> push ebp ; 20/07/2015
2685 00005C9E 56 <1> push esi
2686 00005C9F 57 <1> push edi
2687 00005CA0 53 <1> push ebx
2688 00005CA1 51 <1> push ecx
2689 00005CA2 8B35[AC8E0100] <1> mov esi, [u.pgdir]
2690 00005CA8 89C7 <1> mov edi, eax
2691 00005CAA 50 <1> push eax ; save child's page directory address
2692 <1> ; copy PDE 0 from the parent's page dir to the child's page dir
2693 <1> ; (use same system space for all user page tables)
2694 00005CAB A5 <1> movsd
2695 00005CAC BD00004000 <1> mov ebp, 1024*4096 ; pass the 1st 4MB (system space)
2696 00005CB1 B9FF030000 <1> mov ecx, (PAGE_SIZE / 4) - 1 ; 1023
2697 <1> cpd_0:
2698 00005CB6 AD <1> lodsd
2699 <1> ; or eax, eax
2700 <1> ; jnz short cpd_1
2701 00005CB7 A801 <1> test al, PDE_A_PRESENT ; bit 0 = 1
2702 00005CB9 7508 <1> jnz short cpd_1
2703 <1> ; (virtual address at the end of the page table)
2704 00005CBB 81C500004000 <1> add ebp, 1024*4096 ; page size * PTE count
2705 00005CC1 E80F <1> jmp short cpd_2
2706 <1> cpd_1:
2707 00005CC3 662500F0 <1> and ax, PDE_A_CLEAR ; 0F000h ; clear attribute bits
2708 00005CC7 89C3 <1> mov ebx, eax
2709 <1> ; EBX = Parent's page table address
2710 00005CC9 E81F000000 <1> call copy_page_table
2711 00005CCE 720C <1> jc short cpd_p_err
2712 <1> ; EAX = Child's page table address
2713 00005CD0 0C07 <1> or al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
2714 <1> ; set bit 0, bit 1 and bit 2 to 1
2715 <1> ; (present, writable, user)

```

```

2716      <1> cpd_2:
2717 00005CD2 AB      <1>      stosd
2718 00005CD3 E2E1    <1>      loop   cpd_0
2719      <1>      ;
2720 00005CD5 58      <1>      pop    eax    ; restore child's page directory address
2721      <1> cpd_3:
2722 00005CD6 59      <1>      pop    ecx
2723 00005CD7 5B      <1>      pop    ebx
2724 00005CD8 5F      <1>      pop    edi
2725 00005CD9 5E      <1>      pop    esi
2726 00005CDA 5D      <1>      pop    ebp
2727      <1> cpd_err:
2728 00005CDB C3      <1>      retn
2729      <1> cpd_p_err:
2730      <1>      ; release the allocated pages missing (recover free space)
2731 00005CDC 58      <1>      pop    eax    ; the new page directory address (physical)
2732 00005CDD 8B1D[AC8E0100] <1>      mov    ebx, [u.pgdir] ; parent's page directory address
2733 00005CE3 E86EFCFFFF <1>      call   deallocate_page_dir
2734 00005CE8 29C0    <1>      sub    eax, eax ; 0
2735 00005CEA F9      <1>      stc
2736 00005CEB EBE9    <1>      jmp    short cpd_3
2737      <1>
2738      <1> copy_page_table:
2739      <1>      ; 17/04/2021 (temporary modifications)
2740      <1>      ; 19/09/2015
2741      <1>      ; temporary - 07/09/2015
2742      <1>      ; 07/09/2015 (Retro UNIX 386 v1 - beginning)
2743      <1>      ;
2744      <1>      ; INPUT ->
2745      <1>      ;     EBX = PHYSICAL (real/flat) ADDRESS of the parent's page table.
2746      <1>      ;     EBP = page table entry index (from 'copy_page_dir')
2747      <1>      ; OUTPUT ->
2748      <1>      ;     EAX = PHYSICAL (real/flat) ADDRESS of the child's page table.
2749      <1>      ;     EBP = (recent) page table index (for 'add_to_swap_queue')
2750      <1>      ;     CF = 1 -> error
2751      <1>      ;
2752      <1>      ; Modified Registers -> EBP (except EAX)
2753      <1>      ;
2754 00005CED E834FBFFFF <1>      call   allocate_page
2755 00005CF2 7244    <1>      jc     short cpt_err
2756      <1>      ;
2757 00005CF4 50      <1>      push   eax ; *
2758      <1>      ;push   ebx
2759 00005CF5 56      <1>      push   esi
2760 00005CF6 57      <1>      push   edi
2761 00005CF7 52      <1>      push   edx
2762 00005CF8 51      <1>      push   ecx
2763      <1>      ;
2764 00005CF9 89DE    <1>      mov    esi, ebx
2765 00005CFB 89C7    <1>      mov    edi, eax
2766 00005CFD 89C2    <1>      mov    edx, eax
2767 00005CFF 81C200100000 <1>      add    edx, PAGE_SIZE
2768      <1> cpt_0:
2769      <1>      lodsd
2770 00005D06 A801    <1>      test   al, PTE_A_PRESENT ; bit 0 = 1
2771      <1>      ;jnz    short cpt_1 (*)
2772      <1>      ; 17/04/2021 (temporary) (*)
2773      <1>      ;and    eax, eax (*)
2774 00005D08 741E    <1>      jz     short cpt_2 ; 17/04/2021
2775      <1>      ;
2776      <1>      ; 17/04/2021
2777      <1>      ; ('reload_page' procedure call is disabled as temporary)
2778      <1>      ;
2779      <1>      ; ; ebp = virtual (linear) address of the memory page
2780      <1>      ; call   reload_page ; 28/04/2015
2781      <1>      ; jc     short cpt_p_err
2782      <1> cpt_1:
2783 00005D0A 662500F0 <1>      and    ax, PTE_A_CLEAR ; 0F000h ; clear attribute bits
2784 00005D0E 89C1    <1>      mov    ecx, eax
2785      <1>      ; Allocate a new page for the child process
2786 00005D10 E811FBFFFF <1>      call   allocate_page
2787 00005D15 721C    <1>      jc     short cpt_p_err
2788 00005D17 57      <1>      push   edi
2789 00005D18 56      <1>      push   esi
2790 00005D19 89CE    <1>      mov    esi, ecx
2791 00005D1B 89C7    <1>      mov    edi, eax
2792 00005D1D B900040000 <1>      mov    ecx, PAGE_SIZE/4
2793 00005D22 F3A5    <1>      rep    movsd ; copy page (4096 bytes)
2794 00005D24 5E      <1>      pop    esi
2795 00005D25 5F      <1>      pop    edi
2796      <1>      ;
2797      <1>      ;
2798      <1>      ; 17/04/2021
2799      <1>      ; ('add_to_swap_queue' procedure call is disabled as temporary)
2800      <1>      ;
2801      <1>      ; push   ebx
2802      <1>      ; push   eax
2803      <1>      ; mov    ebx, ebp
2804      <1>      ; ; ebx = virtual address of the memory page
2805      <1>      ; call   add_to_swap_queue
2806      <1>      ; pop    eax
2807      <1>      ; pop    ebx
2808      <1>      ;
2809      <1>      ; or     ax, PTE_A_USER+PTE_A_PRESENT
2810 00005D26 0C07    <1>      or     al, PTE_A_USER+PTE_A_WRITE+PTE_A_PRESENT
2811      <1> cpt_2:
2812 00005D28 AB      <1>      stosd ; EDI points to child's PTE
2813      <1>      ;
2814 00005D29 81C500100000 <1>      add    ebp, 4096 ; 20/07/2015 (next page)
2815      <1>      ;
2816 00005D2F 39D7    <1>      cmp    edi, edx
2817 00005D31 72D2    <1>      jb     short cpt_0
2818      <1> cpt_p_err:
2819 00005D33 59      <1>      pop    ecx
2820 00005D34 5A      <1>      pop    edx
2821 00005D35 5F      <1>      pop    edi
2822 00005D36 5E      <1>      pop    esi
2823      <1>      ;pop   ebx
2824 00005D37 58      <1>      pop    eax ; *
2825      <1> cpt_err:
2826 00005D38 C3      <1>      retn
2827      <1>
2828      <1> allocate_memory_block:
2829      <1>      ; 01/05/2017
2830      <1>      ; 28/04/2017
2831      <1>      ; 25/04/2017
2832      <1>      ; 01/04/2016, 02/04/2016, 03/04/2016
2833      <1>      ; 13/03/2016, 14/03/2016
2834      <1>      ; 12/03/2016 (TRDOS 386 = TRDOS v2.0)
2835      <1>      ; Allocating contiguous memory pages (in the kernel's memory space)
2836      <1>      ;
2837      <1>      ; INPUT ->
2838      <1>      ;     EAX = Beginning address (physical)
2839      <1>      ;     EAX = 0 -> Allocate memory block from the first proper aperture

```



```

2840      <1>      ;      ECX = Number of bytes to be allocated
2841      <1>      ;
2842      <1>      ;      OUTPUT ->
2843      <1>      ;      1) cf = 0 -> successful
2844      <1>      ;      EAX = Beginning (physical) address of the allocated memory block
2845      <1>      ;      ECX = Number of allocated bytes (rounded up to page borders)
2846      <1>      ;      2) cf = 1 -> unsuccessful
2847      <1>      ;      2.1) If EAX > 0 ->
2848      <1>      ;      (Number of requested pages is more than # of free pages
2849      <1>      ;      but contiguous free pages -the aperture- is not enough!)
2850      <1>      ;      EAX = Beginning address of available aperture
2851      <1>      ;      (one of all aperture with max. aperture size/length)
2852      <1>      ;      ECX = Size of available aperture (memory block) in bytes
2853      <1>      ;      2.2) If EAX = 0 -> Out of memory error
2854      <1>      ;      (number of free pages is less than requested number)
2855      <1>      ;      ECX = Total number of free bytes (free pages * 4096)
2856      <1>      ;      (It is not number of contiguous free bytes)
2857      <1>      ;
2858      <1>      ;      (Modified Registers -> EAX, ECX)
2859      <1>      ;
2860      <1>      ;      PURPOSE: Loading a file at memory for copying or running etc.
2861      <1>      ;      If this procedure returns with cf is set, ECX contains maximum
2862      <1>      ;      available space and EAX contains the beginning address of it.
2863      <1>      ;      If EAX has zero, ECX contains total number of free bytes.
2864      <1>      ;      If requested block has been successfully allocated (by rounding up to
2865      <1>      ;      the last page border), it must be deallocated later by using
2866      <1>      ;      'deallocate_memory_block' procedure.
2867      <1>      ;
2868      00005D39 52      <1>      push     edx ; *
2869      00005D3A 8AFF0F0000 <1>      mov     edx, PAGE_SIZE - 1 ; 4095
2870      00005D3F 01D0    <1>      add     eax, edx
2871      00005D41 01D1    <1>      add     ecx, edx
2872      00005D43 C1E90C  <1>      shr     ecx, PAGE_SHIFT ; 12
2873      <1>      ;
2874      <1>      ;      ECX = number of contiguous pages to be allocated
2875      00005D46 8B15[A8760100] <1>      mov     edx, [free_pages]
2876      <1>      ;      01/05/2017
2877      <1>      ;      or     ecx, ecx
2878      <1>      ;      jz     short amb3
2879      <1>      ;      If ECX=0, set cf to 1 and return with max. available mem block size
2880      <1>      ;
2881      00005D4C 39D1    <1>      cmp     ecx, edx
2882      00005D4E 7760    <1>      ja     short amb_3
2883      <1>      ;
2884      00005D50 C1E80C  <1>      shr     eax, PAGE_SHIFT ; 12
2885      <1>      ;
2886      00005D53 89C2    <1>      mov     edx, eax ; page number
2887      00005D55 C1EA03  <1>      shr     edx, 3 ; to get offset to M.A.T.
2888      <1>      ;      (1 allocation bit = 1 page)
2889      <1>      ;      (1 allocation bytes = 8 pages)
2890      00005D58 80E2FC  <1>      and     dl, 0Fch ; clear lower 2 bits
2891      <1>      ;      (to get 32 bit position)
2892      00005D5B 53      <1>      push     ebx ; **
2893      <1>      ;
2894      00005D5C 890D[5C820100] <1>      mov     [mem_ipg_count], ecx ; initial (reset) value of page count
2895      00005D62 890D[60820100] <1>      mov     [mem_pg_count], ecx
2896      00005D68 31C9    <1>      xor     ecx, ecx ; 0
2897      00005D6A 890D[64820100] <1>      mov     [mem_aperture], ecx ; 0
2898      00005D70 890D[68820100] <1>      mov     [mem_max_aperture], ecx ; 0
2899      <1>      ;
2900      00005D76 BB00001000 <1>      mov     ebx, MEM_ALLOC_TBL ; Memory Allocation Table address.
2901      00005D7B 3B15[AC760100] <1>      cmp     edx, [next_page] ; Is the beginning page address lower
2902      <1>      ;      than the address in 'next_page' ?
2903      <1>      ;      (the first/next free page of user space)
2904      00005D81 7208    <1>      jb     short amb_1
2905      00005D83 3B15[B0760100] <1>      cmp     edx, [last_page] ; is the beginning page address higher
2906      <1>      ;      than the address in 'last_page' ?
2907      <1>      ;      (end of the memory)
2908      00005D89 7606    <1>      jna     short amb_2 ; no
2909      <1>      ;
2910      00005D8B 8B15[AC760100] <1>      mov     edx, [next_page] ; M.A.T. offset (1 M.A.T. byte = 8 pages)
2911      <1>      ;
2912      00005D91 01D3    <1>      amb_2: add     ebx, edx
2913      <1>      ;
2914      <1>      ;      28/04/2017
2915      <1>      ;      xor     ecx, ecx
2916      00005D93 0FBC0B  <1>      bsf     ecx, [ebx] ; 0 to 31
2917      00005D96 89D0    <1>      mov     eax, edx
2918      00005D98 C1E003  <1>      shl     eax, 3 ; *8
2919      00005D9B 01C8    <1>      add     eax, ecx ; beginning page number
2920      <1>      ;
2921      00005D9D A3[6C820100] <1>      mov     [mem_pg_pos], eax ; beginning page no (for curr. mem. aperture)
2922      00005DA2 A3[70820100] <1>      mov     [mem_max_pg_pos], eax ; beginning page no for max. mem. aperture
2923      <1>      ;
2924      00005DA7 83E01F  <1>      and     eax, 1Fh ; lower 5 bits only (0 to 31)
2925      <1>      ;      (allocation bit position)
2926      00005DAA 750E    <1>      jnz     short amb_4 ; 0
2927      00005DAC B120    <1>      mov     cl, 32
2928      00005DAE EB4B    <1>      jmp     short amb_10
2929      <1>      ;
2930      <1>      ;      amb_3: ; out_of_memory
2931      00005DB0 31C0    <1>      xor     eax, eax ; 0
2932      00005DB2 89D1    <1>      mov     ecx, edx ; free pages
2933      00005DB4 C1E10C  <1>      shl     ecx, PAGE_SHIFT
2934      00005DB7 5A      <1>      pop     edx ; *
2935      00005DB8 F9      <1>      stc
2936      00005DB9 C3      <1>      retn
2937      <1>      ;
2938      00005DBA 8B13    <1>      amb_4: mov     edx, [ebx]
2939      00005DBC 88C1    <1>      mov     cl, al ; 1 to 31
2940      00005DBE D3EA    <1>      shr     edx, cl
2941      00005DC0 89D0    <1>      mov     eax, edx
2942      <1>      ;
2943      00005DC2 D1E8    <1>      amb_5: shr     eax, 1 ; (***)
2944      00005DC4 7317    <1>      jnc     short amb_7
2945      00005DC6 FF05[64820100] <1>      inc     dword [mem_aperture]
2946      00005DCC FF0D[60820100] <1>      dec     dword [mem_pg_count]
2947      00005DD2 747F    <1>      jz     short amb_15
2948      <1>      ;
2949      <1>      ;      amb_6: ; 28/04/2017
2950      00005DD4 FEC1    <1>      inc     cl
2951      00005DD6 80F920 <1>      cmp     cl, 32
2952      00005DD9 730D    <1>      jnb     short amb_9
2953      00005ddb EB55    <1>      jmp     short amb_5
2954      <1>      ;
2955      00005DDD 50      <1>      amb_7: push     eax ; (***) allocation bits (in shifted status)
2956      00005DDE E828010000 <1>      call    amb_26 ; set maximum memory aperture (free memory block size)
2957      00005DE3 58      <1>      pop     eax ; (***)
2958      00005DE4 EBEE    <1>      jmp     short amb_6
2959      <1>      ;
2960      <1>      ;      amb_8: ; 28/04/2017
2961      00005DE6 B120    <1>      mov     cl, 32
2962      <1>      ;
2963      00005DE8 89DA    <1>      amb_9: mov     edx, ebx

```

```

2964 00005DEA 81EA00001000 <1> sub edx, MEM_ALLOC_TBL
2965 00005DF0 3B15[B0760100] <1> cmp edx, [last_page]
2966 00005DF6 7336 <1> jnb short amb_14 ; contiguous pages not enough
2967 00005DF8 83C304 <1> add ebx, 4
2968 <1> amb_10:
2969 00005DFB 8B03 <1> mov eax, [ebx]
2970 00005DFD 21C0 <1> and eax, eax
2971 00005DFF 7408 <1> jz short amb_11 ; there is not a free page bit in this alloc dword
2972 00005E01 40 <1> inc eax ; 0FFFFFFFh -> 0
2973 00005E02 740C <1> jz short amb_12 ; all of bits are set (32 free pages)
2974 00005E04 48 <1> dec eax
2975 00005E05 28C9 <1> sub cl, cl ; 0
2976 00005E07 EBB9 <1> jmp short amb_5
2977 <1> amb_11:
2978 00005E09 E8FD000000 <1> call amb_26 ; set maximum memory aperture (free memory block size)
2979 00005E0E EBD8 <1> jmp short amb_9
2980 <1> amb_12:
2981 00005E10 390D[60820100] <1> cmp [mem_pg_count], ecx ; 32
2982 00005E16 7306 <1> jnb short amb_13
2983 00005E18 8B0D[60820100] <1> mov ecx, [mem_pg_count]
2984 <1> amb_13:
2985 00005E1E 010D[64820100] <1> add [mem_aperture], ecx
2986 00005E24 290D[60820100] <1> sub [mem_pg_count], ecx
2987 00005E2A 7627 <1> jna short amb_15
2988 00005E2C EBBA <1> jmp short amb_9 ; 01/05/2017
2989 <1> amb_14:
2990 00005E2E E8D8000000 <1> call amb_26 ; 28/04/2017
2991 00005E33 A1[70820100] <1> mov eax, [mem_max_pg_pos] ; begin address of max. mem aperture
2992 00005E38 8B0D[68820100] <1> mov ecx, [mem_max_aperture] ; max. (largest) memory aperture
2993 00005E3E F9 <1> stc
2994 00005E3F E9BC000000 <1> jmp amb_25
2995 <1>
2996 <1> allocate_lfb_memory_block:
2997 <1> ; 28/11/2023
2998 <1> ; 20/10/2023 - TRDOS 386 v2.0.7
2999 <1> ; (short way to set the LFB page bits on the M.A.T.)
3000 <1> ; called from 'allocate_lfb_pages_for_kernel'
3001 00005E44 A3[6C820100] <1> mov [mem_pg_pos], eax ; LFB start/base address as page number
3002 00005E49 890D[64820100] <1> mov [mem_aperture], ecx ; number of LFB pages
3003 <1> ; (!which overlapping main memory!)
3004 00005E4F 52 <1> push edx ; *
3005 00005E50 53 <1> push ebx ; **
3006 00005E51 EB0B <1> jmp short amb_16
3007 <1>
3008 <1> amb_15: ; OK !
3009 00005E53 A1[6C820100] <1> mov eax, [mem_pg_pos] ; Beginning address as page number
3010 00005E58 8B0D[64820100] <1> mov ecx, [mem_aperture] ; Free contiguous page count (>=1)
3011 <1> amb_16:
3012 <1> ; allocate contiguous memory pages (via memory allocation table bits)
3013 00005E5E 89C2 <1> mov edx, eax
3014 <1> ; 25/04/2017
3015 00005E60 C1EA03 <1> shr edx, 3 ; 8 pages in one allocation byte
3016 00005E63 80E2FC <1> and dl, 0Fch ; clear lower 2 bits
3017 <1> ; (for dword/32bit positioning)
3018 <1>
3019 00005E66 BB00001000 <1> mov ebx, MEM_ALLOC_TBL
3020 00005E6B 01D3 <1> add ebx, edx
3021 00005E6D 83E01F <1> and eax, 1Fh ; 31
3022 <1> ; 03/04/2016
3023 00005E70 BA20000000 <1> mov edx, 32
3024 00005E75 28C2 <1> sub dl, al
3025 00005E77 39CA <1> cmp edx, ecx ; ecx >= 1
3026 00005E79 7602 <1> jna short amb_17
3027 00005E7B 89CA <1> mov edx, ecx
3028 <1> amb_17:
3029 00005E7D 29D1 <1> sub ecx, edx
3030 00005E7F 51 <1> push ecx ; ***
3031 00005E80 89D1 <1> mov ecx, edx
3032 <1> amb_18:
3033 00005E82 0FB303 <1> btr [ebx], eax ; The destination bit indexed by the source value
3034 <1> ; is copied into the Carry Flag and then cleared
3035 <1> ; in the destination.
3036 <1> ; 20/10/2023
3037 <1> ; (for LFB page allocation..
3038 <1> ; -here, to prevent using LFB pages for another allocation later-
3039 <1> ; /// the destination bit must be 0 -already- ..
3040 <1> ; so, following 'jnc' is not necessary but i am not removing it
3041 <1> ; for understanding why free page count is decreased here.)
3042 <1> ;
3043 <1> ;jnc short amb_28 ; requested page is already allocated
3044 <1>
3045 00005E85 FF0D[A8760100] <1> dec dword [free_pages] ; 1 page has been allocated (X = X-1)
3046 <1> ;amb_28:
3047 00005E8B 49 <1> dec ecx ; 20/10/2023
3048 00005E8C 7404 <1> jz short amb_19
3049 00005E8E FEC0 <1> inc al
3050 00005E90 EBF0 <1> jmp short amb_18
3051 <1> amb_19:
3052 00005E92 59 <1> pop ecx ; ***
3053 <1> ;and ecx, ecx ; 0 ?
3054 <1> ;jz short amb_22
3055 <1> ; 28/11/2023
3056 00005E93 E31E <1> jecxz amb_22
3057 <1> ; 01/04/2016
3058 00005E95 B020 <1> mov al, 32
3059 <1> amb_20:
3060 00005E97 83C304 <1> add ebx, 4
3061 00005E9A 39C1 <1> cmp ecx, eax ; 32
3062 00005E9C 7305 <1> jnb short amb_21
3063 <1> ; ECX < 32
3064 00005E9E 28C0 <1> sub al, al ; 0
3065 00005EA0 50 <1> push eax ; 0 ***
3066 00005EA1 EBD8 <1> jmp short amb_18
3067 <1> amb_21:
3068 00005EA3 2905[A8760100] <1> sub [free_pages], eax ; [free_pages] = [free_pages] - 32
3069 00005EA9 C70300000000 <1> mov dword [ebx], 0 ; reset 32 bits
3070 00005EAF 29C1 <1> sub ecx, eax ; 32
3071 00005EB1 75E4 <1> jnz short amb_20
3072 <1> amb_22:
3073 00005EB3 A1[6C820100] <1> mov eax, [mem_pg_pos] ; Beginning address as page number
3074 00005EB8 8B0D[64820100] <1> mov ecx, [mem_aperture] ; Free contiguous page count
3075 <1> ; [next_page] update
3076 00005EBE 89C2 <1> mov edx, eax
3077 <1> ; 03/04/2016
3078 00005EC0 C1EA03 <1> shr edx, 3 ; to get offset to M.A.T.
3079 <1> ; (1 allocation bit = 1 page)
3080 <1> ; (1 allocation bytes = 8 pages)
3081 00005EC3 80E2FC <1> and dl, 0Fch ; clear lower 2 bits
3082 <1> ; (to get 32 bit position)
3083 00005EC6 3B15[AC760100] <1> cmp edx, [next_page] ; first free page pointer offset
3084 00005ECC 7732 <1> ja short amb_25
3085 00005ECE BB00001000 <1> mov ebx, MEM_ALLOC_TBL
3086 00005ED3 833C1300 <1> cmp dword [ebx+edx], 0
3087 00005ED7 7721 <1> ja short amb_24

```

```

3088 00005ED9 89C2      <1>    mov     edx, eax
3089 00005EDB 01CA      <1>    add     edx, ecx
3090 00005EDD C1EA03    <1>    shr     edx, 3
3091 00005EE0 80E2FC    <1>    and     dl, 0Fch
3092                                <1> amb_23:
3093 00005EE3 833C1300    <1>    cmp     dword [ebx+edx], 0
3094 00005EE7 7711      <1>    ja      short amb_24
3095 00005EE9 83C204    <1>    add     edx, 4
3096 00005EEC 3B15[B0760100] <1>    cmp     edx, [last_page] ; last page pointer offset
3097 00005EF2 76EF      <1>    jna     short amb_23
3098 00005EF4 8B15[B4760100] <1>    mov     edx, [first_page] ; (for) beginning of user's space
3099                                <1> amb_24:
3100 00005EFA 8915[AC760100] <1>    mov     [next_page], edx
3101                                <1> amb_25:
3102 00005F00 9C      <1>    pushf
3103 00005F01 C1E00C    <1>    shl     eax, PAGE_SHIFT ; convert to phy. address in bytes
3104 00005F04 C1E10C    <1>    shl     ecx, PAGE_SHIFT ; convert to byte counts
3105 00005F07 9D      <1>    popf
3106 00005F08 5B      <1>    pop     ebx ; **
3107 00005F09 5A      <1>    pop     edx ; *
3108 00005F0A C3      <1>    retn
3109                                <1>
3110                                <1> amb_26: ; set maximum free memory aperture (free memory block size)
3111 00005F0B 89DA      <1>    mov     edx, ebx ; current address
3112 00005F0D 81EA00001000 <1>    sub     edx, MEM_ALLOC_TBL ; MAT beginning address
3113                                <1> ; 02/04/2016
3114 00005F13 C1E203    <1>    shl     edx, 3 ; MAT byte offset * 8 = page number base
3115 00005F16 01CA      <1>    add     edx, ecx ; current page number (ecx = 0 to 32)
3116                                <1> ;
3117 00005F18 A1[64820100] <1>    mov     eax, [mem_aperture]
3118 00005F1D 21C0      <1>    and     eax, eax
3119 00005F1F 7421      <1>    jz      short amb_27
3120 00005F21 C705[64820100]0000- <1>    mov     dword [mem_aperture], 0
3121 00005F29 0000      <1>
3122 00005F2B 3B05[68820100] <1>    cmp     eax, [mem_max_aperture]
3123 00005F31 760F      <1>    jna     short amb_27
3124 00005F33 A3[68820100] <1>    mov     [mem_max_aperture], eax
3125                                <1> ; 25/04/2017
3126 00005F38 A1[6C820100] <1>    mov     eax, [mem_pg_pos]
3127 00005F3D A3[70820100] <1>    ; EAX = Beginning page number of the max. aperture
3128                                <1> mov     [mem_max_pg_pos], eax
3129                                <1> amb_27:
3130 00005F42 8915[6C820100] <1>    mov     [mem_pg_pos], edx ; current page
3131                                <1>
3132 00005F48 A1[5C820100] <1>    mov     eax, [mem_ipg_count] ; initial (reset) value of page count
3133 00005F4D A3[60820100] <1>    mov     [mem_pg_count], eax
3134 00005F52 C3      <1>    retn
3135                                <1>
3136                                <1> ; 04/06/2024
3137                                <1> %if 0
3138                                <1> deallocate_memory_block_x:
3139                                <1> ; 26/11/2023 - TRDOS 386 v2.0.7
3140                                <1> ; deallocate pages after rounding up
3141                                <1> ; ((audio buffer size etc. may not be rounded up to page boundary))
3142                                <1> add     ecx, PAGE_SIZE - 1 ; 4095
3143                                <1> %endif
3144                                <1>
3145                                <1> deallocate_memory_block:
3146                                <1> ; 03/04/2016
3147                                <1> ; 14/03/2016 (TRDOS 386 = TRDOS v2.0)
3148                                <1> ; Deallocating contiguous memory pages (in the kernel's memory space)
3149                                <1> ;
3150                                <1> ; INPUT ->
3151                                <1> ; EAX = Beginning address (physical)
3152                                <1> ; ECX = Number of bytes to be deallocated
3153                                <1> ;
3154                                <1> ; OUTPUT ->
3155                                <1> ; Memory Allocation Table bits will be updated
3156                                <1> ; [free_pages] will be changed (increased)
3157                                <1> ;
3158                                <1> ; (Modified Registers -> EAX, ECX)
3159                                <1> ;
3160                                <1> ; PURPOSE: Unloading/Freeing a file -or an allocated memory block-
3161                                <1> ; at memory after copying, running, saving, reading, writing etc.
3162                                <1> ;
3163                                <1> ;
3164 00005F53 52      <1>    push     edx ; *
3165 00005F54 53      <1>    push     ebx ; **
3166                                <1>
3167 00005F55 C1E80C    <1>    shr     eax, PAGE_SHIFT ; 12
3168 00005F58 C1E90C    <1>    shr     ecx, PAGE_SHIFT ; 12
3169                                <1>
3170                                <1> ; EAX = Beginning page number
3171                                <1> ; ECX = Number of contiguous pages to be deallocated
3172                                <1> damb_0:
3173                                <1> ; deallocate contiguous memory pages (via memory allocation table bits)
3174 00005F5B 89C2      <1>    mov     edx, eax
3175 00005F5D C1EA03    <1>    shr     edx, 3 ; to get offset to M.A.T.
3176                                <1> ; (1 allocation bit = 1 page)
3177                                <1> ; (1 allocation bytes = 8 pages)
3178 00005F60 80E2FC    <1>    and     dl, 0Fch ; clear lower 2 bits
3179                                <1> ; (to get 32 bit position)
3180 00005F63 3B15[AC760100] <1>    cmp     edx, [next_page] ; next free page
3181 00005F69 7306      <1>    jnb     short damb_1
3182 00005F6B 8915[AC760100] <1>    mov     [next_page], edx
3183                                <1> damb_1:
3184 00005F71 8B00001000 <1>    mov     ebx, MEM_ALLOC_TBL
3185 00005F76 01D3      <1>    add     ebx, edx
3186 00005F78 83E01F    <1>    and     eax, 1Fh ; 31
3187                                <1>
3188                                <1> ; 03/04/2016
3189 00005F7B BA20000000 <1>    mov     edx, 32
3190 00005F80 28C2      <1>    sub     dl, al
3191 00005F82 39CA      <1>    cmp     edx, ecx
3192 00005F84 7602      <1>    jna     short damb_2
3193 00005F86 89CA      <1>    mov     edx, ecx
3194                                <1> damb_2:
3195 00005F88 29D1      <1>    sub     ecx, edx
3196 00005F8A 51      <1>    push     ecx ; ***
3197 00005F8B 89D1      <1>    mov     ecx, edx
3198                                <1> damb_3:
3199 00005F8D 0FAB03    <1>    bts     [ebx], eax ; unlink/release/deallocate page
3200                                <1> ; set relevant bit to 1.
3201                                <1> ; set CF to the previous bit value
3202 00005F90 FF05[A8760100] <1>    inc     dword [free_pages] ; 1 page has been deallocated (X = x+1)
3203 00005F96 49      <1>    dec     ecx
3204 00005F97 7404      <1>    jz      short damb_4
3205 00005F99 FEC0      <1>    inc     al
3206 00005F9B EBF0      <1>    jmp     short damb_3
3207                                <1> damb_4:
3208 00005F9D 59      <1>    pop     ecx ; ***
3209 00005F9E 21C9      <1>    and     ecx, ecx ; 0 ?
3210 00005FA0 741E      <1>    jz      short damb_7

```

```

3211      ; 03/04/2016
3212 00005FA2 B020      <1>      mov     al, 32
3213      <1>      damb_5:
3214      <1>      add     ebx, 4
3215 00005FA7 39C1      <1>      cmp     ecx, eax ; 32
3216 00005FA9 7305      <1>      jnb     short damb_6
3217      <1>      ; ECX < 32
3218 00005FAB 28C0      <1>      sub     al, al ; 0
3219 00005FAD 50        <1>      push    eax ; 0 ***
3220 00005FAE EBDD      <1>      jmp     short damb_3
3221      <1>      damb_6:
3222 00005FB0 0105[A8760100] <1>      add     [free_pages], eax ; [free_pages] = [free_pages] + 32
3223 00005FB6 C703FFFFFFFF <1>      mov     dword [ebx], 0FFFFFFFFh ; set 32 bits
3224 00005FBC 29C1      <1>      sub     ecx, eax ; 32
3225 00005FBE 75E4      <1>      jnz     short damb_5
3226      <1>      damb_7:
3227 00005FC0 5B        <1>      pop     ebx ; **
3228 00005FC1 5A        <1>      pop     edx ; *
3229 00005FC2 C3        <1>      retn
3230      <1>
3231      <1>      direct_memory_access:
3232      <1>      ; 20/10/2023
3233      <1>      ; 17/04/2021 (temporary modifications)
3234      <1>      ; 22/07/2017
3235      <1>      ; 12/05/2017
3236      <1>      ; 16/07/2016
3237      <1>      ; 12/07/2016 (TRDOS 386 = TRDOS v2.0)
3238      <1>      ; This procedure will be called to map
3239      <1>      ; user's (ring 3) page tables to access physical
3240      <1>      ; (flat/linear) memory addresses, directly (without
3241      <1>      ; kernel's data transfer functions).
3242      <1>
3243      <1>      ; Purpose: Video memory access and shared memory access.
3244      <1>
3245      <1>      INPUT ->
3246      <1>      EAX = Beginning address (physical).
3247      <1>      EBX = User's buffer address ; 12/05/2017
3248      <1>      ECX = Number of contiguous pages to be mapped.
3249      <1>      OUTPUT ->
3250      <1>      User's page directory and pages tables
3251      <1>      will be updated.
3252      <1>
3253      <1>      ; If an old page table entry has valid page address,
3254      <1>      ; that page will be deallocated just before PTE will
3255      <1>      ; be changed for direct (1 to 1) memory page access.
3256      <1>
3257      <1>      ; If old PTE value points to a swapped page,
3258      <1>      ; that page (block) will be unlinked on swap disk.
3259      <1>
3260      <1>      ; Newly allocated pages (except page tables) will not
3261      <1>      ; be applied to Memory Allocation Table.
3262      <1>      ; AVL bit 1 (PTE bit 10) of page table entry will be
3263      <1>      ; used to indicate shared (direct) memory page; then,
3264      <1>      ; this page will not be deallocated later during
3265      <1>      ; process termination. (Memory Allocation Table and
3266      <1>      ; free memory count will not be affected.
3267      <1>      ; (Except deallocating page table's itself.)
3268      <1>
3269      <1>      ; CF = 1 -> error (EAX = error code)
3270      <1>      ; CF = 0 -> success (EAX = beginning address)
3271      <1>
3272      <1>      ; (Modified Registers -> none)
3273      <1>      ; Modified registers: ebp, edx, ecx, ebx, esi, edi
3274      <1>
3275      <1>
3276      <1>      ;push    ebp
3277      <1>      ;push    ebx
3278      <1>      ;push    ecx
3279      <1>      ;push    edx
3280 00005FC3 662500F0 <1>      and     ax, PTE_A_CLEAR ; clear page offset
3281 00005FC7 50        <1>      push    eax
3282      <1>      ;and     ecx, ecx ; page count
3283      <1>      ;jz      dmem_acc_7 ; 'insufficient memory' error
3284 00005FC8 89C5      <1>      mov     ebp, eax
3285 00005FCA 81C300004000 <1>      add     ebx, CORE ; 12/05/2017
3286      <1>      dmem_acc_0:
3287 00005FD0 891D[108D0100] <1>      mov     [base_addr], ebx ; 12/05/2017
3288 00005FD6 A1[AC8E0100] <1>      mov     eax, [u.pgdir] ; page dir address (physical)
3289 00005FDB E858F9FFFF <1>      call    get_pte
3290      <1>      ; EDX = Page table entry address (if CF=0)
3291      <1>      ; Page directory entry address (if CF=1)
3292      <1>      ; (Bit 0 value is 0 if PT is not present)
3293      <1>      ; EAX = Page table entry value (page address)
3294      <1>      ; CF = 1 -> PDE not present or invalid ?
3295 00005FE0 7321      <1>      jnc     short dmem_acc_1
3296      <1>      ; 20/10/2023
3297      <1>      ; Allocate a page as a new page table
3298 00005FE2 E83FF8FFFF <1>      call    allocate_page
3299      <1>      ;jc      dmem_acc_7 ; 'insufficient memory' error
3300      <1>      ; 17/04/2021
3301 00005FE7 7215      <1>      jc      short _dmem_acc_7
3302      <1>
3303      <1>      ;call    clear_page
3304      <1>      ; EAX = Physical (base) address of the allocated (new) page
3305 00005FEE 0C07      <1>      or      al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER ; 4+2+1 = 7
3306      <1>      ; lower 3 bits are used as U/S, R/W, P flags
3307      <1>      ; (user, writable, present page)
3308 00005FF0 8902      <1>      mov     [edx], eax ; Let's put the new page directory entry here !
3309 00005FF2 A1[AC8E0100] <1>      mov     eax, [u.pgdir]
3310 00005FF7 E83CF9FFFF <1>      call    get_pte
3311      <1>      ;jc      dmem_acc_7 ; 'insufficient memory' error
3312      <1>      ; 17/04/2021
3313 00005FFC 7305      <1>      jnc     short dmem_acc_1
3314      <1>      _dmem_acc_7:
3315      <1>      jmp     dmem_acc_7
3316      <1>      dmem_acc_1:
3317      <1>      ; EAX = PTE value, EDX = PTE address
3318 00006003 A801      <1>      test    al, PTE_A_PRESENT
3319      <1>      ;jnz     short dmem_acc_2 ; 17/04/2021 (*)
3320      <1>      ; 17/04/2021 (temporary)
3321 00006005 745F      <1>      jz      short short dmem_acc_6 ; ! temporary ! (*)
3322      <1>      ; 20/10/2023
3323      <1>      ; If the page table entry is zero or invalid -not present-
3324      <1>      ; it will be directly allocated for the physical memory section
3325      <1>      ; without using the M.A.T. (Specially for LFB access -generally-
3326      <1>      ; is at the beyond of the physical main memory end.)
3327      <1>      ; (M.A.T bits are not used for direct memory access. To cancel
3328      <1>      ; direct memory access also is out of the M.A.T scope because
3329      <1>      ; direct accessed physical pages are marked as SHARED on their
3330      <1>      ; PTE. Shared pages are not deallocated.)
3331      <1>
3332      <1>      ; 17/04/2021
3333      <1>      ; (following code is disabled as temporary)
3334      <1>

```

```

3335 <1> ; or     eax, eax
3336 <1> ; jz     short dmem_acc_6 ; Change PTE
3337 <1> ; shr    eax, 1 ; swap disk block (8 sectors) address
3338 <1> ; ; unlink swap disk block
3339 <1> ; call   unlink_swap_block
3340 <1> ; jmp     short dmem_acc_6
3341 <1>
3342 <1> dmem_acc_2:
3343 <1> test    al, PTE_A_WRITE ; bit 1, writable (r/w) flag
3344 <1> ; (must be 1)
3345 <1> jnz     short dmem_acc_4
3346 <1> ; Read only -duplicated- page (belongs to a parent or a child)
3347 <1> test    ax, PTE_DUPLICATED ; was this page duplicated
3348 <1> ; as child's page ?
3349 <1> jz      short dmem_acc_5 ; Change PTE but don't deallocate the page!
3350 <1>
3351 <1> ;push    edi
3352 <1> ;push    esi
3353 <1>
3354 <1> push    ecx
3355 <1> ;push    ebx
3356 <1> mov     ebx, [u.ppgdir] ; parent's page dir address (physical)
3357 <1>
3358 <1> ; check the parent's PTE value is read only & same page or not..
3359 <1> mov     edi, ebp
3360 <1> shr     edi, PAGE_D_SHIFT ; 22
3361 <1> ; EDI = page directory entry index (0-1023)
3362 <1> mov     esi, ebp
3363 <1> shr     esi, PAGE_SHIFT ; 12
3364 <1> and     esi, PTE_MASK
3365 <1> ; ESI = page table entry index (0-1023)
3366 <1>
3367 <1> shl     di, 2 ; * 4
3368 <1> add     ebx, edi ; PDE offset (for the parent)
3369 <1> mov     ecx, [edi]
3370 <1> test    cl, PDE_A_PRESENT ; present (valid) or not ?
3371 <1> jz      short dmem_acc_3 ; parent process does not use this page
3372 <1> and     cx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3373 <1> shl     si, 2 ; *4
3374 <1> add     esi, ecx ; PTE offset (for the parent)
3375 <1> mov     ebx, [esi]
3376 <1> test    bl, PTE_A_PRESENT ; present or not ?
3377 <1> jz      short dmem_acc_3 ; parent process does not use this page
3378 <1> and     ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3379 <1> and     bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3380 <1> cmp     eax, ebx ; parent's and child's pages are same ?
3381 <1> jne     short dmem_acc_3 ; not same page
3382 <1> ; deallocate the child's page
3383 <1> or      byte [esi], PTE_A_WRITE ; convert to writable page (parent)
3384 <1> ;pop     ebx
3385 <1> ;pop     ecx
3386 <1> jmp     short dmem_acc_5
3387 <1> dmem_acc_3:
3388 <1> ;pop     ebx
3389 <1> ;pop     ecx
3390 <1> dmem_acc_4:
3391 <1> test    ax, PTE_SHARED ; shared or direct memory access indicator
3392 <1> jnz     short dmem_acc_5 ; AVL bit 1 = 1, do not deallocate this page!
3393 <1> ;
3394 <1> ;and     ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3395 <1> call    deallocate_page
3396 <1> dmem_acc_5:
3397 <1> ;pop     esi
3398 <1> ;pop     edi
3399 <1> dmem_acc_6:
3400 <1> mov     eax, ebp ; physical page (offset=0) address
3401 <1> ; EAX = memory page address
3402 <1> ; EDX = PTE entry address (physical)
3403 <1> or      ax, PTE_A_PRESENT+PTE_A_USER+PTE_A_WRITE+PTE_SHARED
3404 <1> ; present flag, bit 0 = 1
3405 <1> ; user flag, bit 2 = 1
3406 <1> ; writable flag, bit 1 = 1
3407 <1> ; direct memory access flag, bit 10 = 1
3408 <1> ; (This page must not be deallocated!)
3409 <1> mov     [edx], eax ; Update PTE value
3410 <1> dec     ecx ; remain count of contiguous pages
3411 <1> jz      short dmem_acc_8
3412 <1> add     ebp, PAGE_SIZE ; next physical page address
3413 <1> ; 22/07/2017
3414 <1> ;mov     eax, ebp
3415 <1> ; 12/05/2017
3416 <1> mov     ebx, [base_addr] ; linear address (virtual+CORE)
3417 <1> add     ebx, PAGE_SIZE ; next linear address
3418 <1> jmp     dmem_acc_0
3419 <1> dmem_acc_7: ; ERROR !
3420 <1> mov     dword [esp], ERR_MINOR_IM
3421 <1> ; Insufficient memory (minor) error!
3422 <1> ; Major error = 0 (No protection fault)
3423 <1> ; cf = 1
3424 <1> dmem_acc_8:
3425 <1> pop     eax
3426 <1> ;pop     edx
3427 <1> ;pop     ecx
3428 <1> ;pop     ebx
3429 <1> ;pop     ebp
3430 <1> retn
3431 <1>
3432 <1> deallocate_user_pages:
3433 <1> ; 20/05/2017
3434 <1> ; 15/05/2017
3435 <1> ; 20/02/2017
3436 <1> ; 19/02/2017 (TRDOS 386 = TRDOS v2.0)
3437 <1> ;
3438 <1> ; Deallocate virtually contiguous user pages (memory block)
3439 <1> ; (caller: 'sysdalloc' system call)
3440 <1> ;
3441 <1> ; INPUT ->
3442 <1> ; EBX = VIRTUAL ADDRESS (beginning address)
3443 <1> ; ECX = byte count
3444 <1> ; [u.pgdir] = user's page directory
3445 <1> ; [u.ppgdir] = parent's page directory
3446 <1> ;
3447 <1> ; OUTPUT ->
3448 <1> ; If CF = 0
3449 <1> ; EAX = Deallocated memory bytes
3450 <1> ; (Even if shared or read only pages will not be
3451 <1> ; deallocated on M.A.T., this byte count will be
3452 <1> ; returned as virtually deallocated bytes; in fact
3453 <1> ; virtually deallocated user pages * 4096.)
3454 <1> ; EBX = Virtual address (as rounded up)
3455 <1> ; If CF = 1
3456 <1> ; EAX = 0 (there is not any deallocated pages)
3457 <1> ;
3458 <1> ; Note: Empty page tables will not be deallocated!!!

```

```

3459      ;      (they will be deallocated at process termination stage)
3460      ;
3461      ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP
3462      ;
3463      mov     esi, ebx
3464      mov     edi, esi
3465      add     edi, ecx
3466      add     esi, PAGE_SIZE - 1 ; 4095 (round up)
3467      shr     esi, PAGE_SHIFT
3468      shr     edi, PAGE_SHIFT
3469      mov     eax, edi ; end page
3470      sub     eax, esi ; end page - start page
3471      jna     da_u_pd_err ; < 1
3472      mov     ebx, esi
3473      shl     ebx, PAGE_SHIFT ; virtual address (as rounded up)
3474      push    ebx ; *
3475      mov     ecx, eax ; page count
3476      shl     eax, PAGE_SHIFT ; byte count as adjusted
3477      push    eax ; **
3478      mov     ebx, [u.pgdir] ; physical addr of user's page dir
3479      add     esi, CORE/PAGE_SIZE
3480      mov     edi, esi
3481      and     edi, PTE_MASK ; PTE entry in the page table
3482      push    edi ; *** ; PTE index (of page directory)
3483      shr     esi, PAGE_D_SHIFT - PAGE_SHIFT ; 22-12=10
3484      mov     edx, esi
3485      ; EDX = PDE index
3486      shl     esi, 2 ; convert PDE index to dword offset
3487      add     esi, ebx ; add page directory address
3488      da_u_pd_1:
3489      lodsd
3490      ;
3491      mov     ebp, esi ; 20/02/2017
3492      ; EBP = next PDE address
3493      ;
3494      test    al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
3495      jz      da_u_pd_3 ; 20/05/2017
3496      and     ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3497      ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE
3498      mov     edi, [esp] ; ***
3499      ; EDI = PTE index (of complete page directory)
3500      and     edi, PTE_MASK ; PTE entry in the page table
3501      shl     edi, 2 ; convert PTE index to dword offset
3502      mov     esi, edi ; PTE offset in page table (0-4092)
3503      add     esi, eax ; now, esi points to requested PTE
3504      da_u_pt_0:
3505      lodsd
3506      test    al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
3507      jz      short da_u_pt_1
3508      ;
3509      test    al, PTE_A_WRITE ; bit 1, writable (r/w) flag
3510      ; (must be 1)
3511      jnz     short da_u_pt_3
3512      ; Read only -duplicated- page (belongs to a parent or a child)
3513      test    ax, PTE_DUPLICATED ; was this page duplicated
3514      ; as child's page ?
3515      jz      short da_u_pt_4 ; Clear PTE but don't deallocate the page!
3516      ;
3517      ; check the parent's PTE value is read only & same page or not..
3518      ; EDX = page directory entry index (0-1023)
3519      push    edx ; ****
3520      ; EDI = page table entry offset (0-4092)
3521      mov     ebx, [u.pgdir] ; page directory of the parent process
3522      shl     dx, 2 ; *4
3523      add     ebx, edx ; PDE address (for the parent)
3524      mov     edx, [ebx] ; page table address
3525      test    dl, PDE_A_PRESENT ; present (valid) or not ?
3526      jz      short da_u_pt_2 ; parent process does not use this page
3527      and     dx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3528      ; EDI = page table entry offset (0-4092)
3529      add     edi, edx ; PTE address (for the parent)
3530      mov     ebx, [edi]
3531      test    bl, PTE_A_PRESENT ; present or not ?
3532      jz      short da_u_pt_2 ; parent process does not use this page
3533      and     ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3534      and     bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3535      cmp     eax, ebx ; parent's and child's pages are same ?
3536      jne     short da_u_pt_2 ; not same page
3537      ; deallocate the child's page
3538      or      byte [edi], PTE_A_WRITE ; convert to writable page (parent)
3539      pop     edx ; ****
3540      jmp     short da_u_pt_4
3541      ;
3542      ; 17/04/2021
3543      ; ('da_u_pt_1' is disabled as temporary)
3544      ;
3545      da_u_pt_1:
3546      ; or      eax, eax ; swapped page ?
3547      jz      short da_u_pt_5 ; no
3548      ; yes
3549      ; shr     eax, 1
3550      call    unlink_swap_block ; Deallocate swapped page block
3551      ; on the swap disk (or in file)
3552      ; jmp     short da_u_pt_5
3553      da_u_pt_2:
3554      pop     edx ; ****
3555      da_u_pt_3:
3556      test    ax, PTE_SHARED ; shared or direct memory access indicator
3557      jnz     short da_u_pt_4 ; AVL bit 1 = 1, do not deallocate this page!
3558      ;
3559      ; and     ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3560      call    deallocate_page ; set the mem allocation bit of this page
3561      da_u_pt_4:
3562      mov     dword [esi-4], 0 ; clear/reset PTE (child, dupl. as parent)
3563      ; 17/04/2021 (temporary)
3564      da_u_pt_1:
3565      da_u_pt_5:
3566      ; 20/05/2017
3567      pop     eax ; *** PTE index (of page directory)
3568      dec     ecx ; remain page count
3569      jz      short da_u_pd_4
3570      inc     eax ; next PTE
3571      and     ax, PTE_MASK ; PTE entry index in the page table
3572      push    eax ; *** (save again)
3573      ; mov     edi, eax
3574      ; and     di, PTE_MASK
3575      ; cmp     edi, PAGE_SIZE / 4 ; 1024
3576      ; jnb     short da_u_pd_2
3577      mov     edi, eax
3578      shl     edi, 2 ; convert index to dword offset
3579      ; test    ax, PTE_MASK ; 3FFh
3580      or      eax, eax
3581      jnz     short da_u_pt_0 ; 1-1023
3582      da_u_pd_2:

```

```

3583 0000615B 42      <1>      inc     edx
3584                <1>      ; 20/05/2017
3585 0000615C 6681E2FF03 <1>      and     dx, PTE_MASK ; 3FFh
3586 00006161 740F      <1>      jz      short da_u_pd_4 ; 0 (1024)
3587                <1>      ;cmp     edx, 1024
3588                <1>      ;jnb     short da_u_pd_4
3589 00006163 89EE      <1>      mov     esi, ebp ; 20/02/2017
3590 00006165 E96EFFFF    <1>      jmp     da_u_pd_1
3591                <1>      da_u_pd_3:
3592                <1>      ; 15/05/2017 (empty page directory entry)
3593 0000616A 81E900040000 <1>      sub     ecx, 1024
3594 00006170 77E9      <1>      ja      short da_u_pd_2 ; 20/05/2017
3595                <1>      da_u_pd_4:
3596 00006172 58      <1>      pop     eax ; **
3597 00006173 5B      <1>      pop     ebx ; *
3598 00006174 C3      <1>      retn
3599                <1>
3600                <1>      da_u_pd_err:
3601 00006175 31C0      <1>      xor     eax, eax
3602 00006177 F9      <1>      stc
3603 00006178 C3      <1>      retn
3604                <1>
3605                <1>      allocate_user_pages:
3606                <1>      ; 20/05/2017
3607                <1>      ; 01/05/2017, 02/05/2017, 15/05/2017
3608                <1>      ; 04/03/2017
3609                <1>      ; 20/02/2017 (TRDOS 386 = TRDOS v2.0)
3610                <1>
3611                <1>      ; Allocate physically contiguous user pages (memory block)
3612                <1>      ; (caller: 'sysalloc' system call)
3613                <1>
3614                <1>      ; Note: This procedure does not alloc a page's itself
3615                <1>      ; (page bits) on Memory Allocation Table.
3616                <1>      ; (allocate_memory_block is needed before this proc)
3617                <1>
3618                <1>      INPUT ->
3619                <1>      ; EAX = PHYSICAL ADDRESS (beginning address)
3620                <1>      ; EBX = VIRTUAL ADDRESS (beginning address)
3621                <1>      ; ECX = byte count (>=4096)
3622                <1>      ; [u.pgdir] = user's page directory
3623                <1>
3624                <1>      ; Note: All addresses are (must be) already adjusted
3625                <1>      ; to page borders, otherwise, lower 12bits of addresses
3626                <1>      ; and byte count would be truncated.
3627                <1>
3628                <1>      OUTPUT ->
3629                <1>      ; none
3630                <1>
3631                <1>      ; CF = 1 -> insufficient memory error
3632                <1>
3633                <1>      ; Note: All pages will be allocated in physical page order
3634                <1>      ; from the beginning page address.
3635                <1>      ; * A new page table will be added to the page dir
3636                <1>      ; when the requested PDE is invalid.
3637                <1>      ; * Those pages will not be added to swap queue
3638                <1>      ; because main purpose of this allocation is to
3639                <1>      ; set a direct memory access (DMA controller) buffer.
3640                <1>      ; (Swapping out a page in a DMA buffer would be wrong!)
3641                <1>      ; * Previous content of page tables (PTEs) would be
3642                <1>      ; (should be) deallocated before entering this
3643                <1>      ; procedure. So, new page table entries (PTEs)
3644                <1>      ; directly will be written without checking
3645                <1>      ; their previous content.
3646                <1>      ; * Only solution to increase free memory by removing
3647                <1>      ; that non-swappable memory block is to terminate
3648                <1>      ; the process or to wait until the process will
3649                <1>      ; deallocate that memory block as itself. ('sysdalloc')
3650                <1>      ; (No problem, if the process does not grab all of
3651                <1>      ; -very big amount of- free memory by using
3652                <1>      ; 'sysalloc' system call!?)
3653                <1>      ; (Even if the process has grabbed all of free memory,
3654                <1>      ; no problem if the process is not running in
3655                <1>      ; multitasking mode. No problem in multitasking
3656                <1>      ; mode if there is not another process which is running
3657                <1>      ; or waiting or sleeping for an event as it's pages
3658                <1>      ; are swapped-out. But a new process can not start to
3659                <1>      ; run if all of free memory has been allocated
3660                <1>      ; by running processes. Deallocation -'sysdalloc'-
3661                <1>      ; or terminate a running process is needed
3662                <1>      ; in order to run a new process.)
3663                <1>
3664                <1>      ; Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP
3665                <1>
3666                <1>
3667                <1>      ; 01/05/2017
3668 00006179 662500F0      <1>      and     ax, ~PAGE_OFF
3669 0000617D 6681E300F0 <1>      and     bx, ~PAGE_OFF
3670                <1>      ; 02/05/2017
3671 00006182 BD00F0FFFF    <1>      mov     ebp, 0FFFFFF00h ; 4 Giga Bytes - 4096 Bytes (for Stack)
3672 00006187 C1E90C      <1>      shr     ecx, PAGE_SHIFT ; page count
3673 0000618A 83F901      <1>      cmp     ecx, 1
3674 0000618D 7251      <1>      jb      short a_u_im_retn
3675 0000618F 89C2      <1>      mov     edx, eax
3676 00006191 01CA      <1>      add     edx, ecx
3677 00006193 724B      <1>      jc      short a_u_im_retn
3678 00006195 39D5      <1>      cmp     ebp, edx
3679 00006197 7247      <1>      jb      short a_u_im_retn
3680 00006199 89DA      <1>      mov     edx, ebx
3681 0000619B 81C200004000 <1>      add     edx, CORE
3682 000061A1 723D      <1>      jc      short a_u_im_retn
3683 000061A3 01CA      <1>      add     edx, ecx
3684 000061A5 7239      <1>      jc      short a_u_im_retn
3685 000061A7 39D5      <1>      cmp     ebp, edx
3686 000061A9 7235      <1>      jb      short a_u_im_retn
3687                <1>
3688 000061AB 89C5      <1>      mov     ebp, eax ; physical address
3689 000061AD 89DE      <1>      mov     esi, ebx
3690 000061AF 81C600004000 <1>      add     esi, CORE ; start of user's memory (4M)
3691 000061B5 C1EE0C      <1>      shr     esi, PAGE_SHIFT ; higher 20 bits of the linear address
3692                <1>      ;shr     ecx, PAGE_SHIFT ; page count
3693 000061B8 8B1D[AC8E0100] <1>      mov     ebx, [u.pgdir] ; physical addr of user's page dir
3694 000061BE 89F7      <1>      mov     edi, esi
3695 000061C0 81E7FF030000 <1>      and     edi, PTE_MASK ; PTE entry index in the page table
3696 000061C6 57      <1>      push    edi ; * ; PTE index (in page directory)
3697 000061C7 C1EE0A      <1>      shr     esi, PAGE_D_SHIFT - PAGE_SHIFT ; 22-12=10
3698 000061CA 89F2      <1>      mov     edx, esi
3699                <1>      ; EDX = PDE index
3700 000061CC C1E602      <1>      shl     esi, 2 ; convert PDE index to dword offset
3701 000061CF 01DE      <1>      add     esi, ebx ; add page directory address
3702                <1>      a_u_pd_0:
3703 000061D1 AD      <1>      lodsd
3704                <1>
3705 000061D2 89F3      <1>      mov     ebx, esi ; next PDE address
3706                <1>

```

```

3707 000061D4 A801      <1> test    al, PDE_A_PRESENT ; bit 0, present flag (must be 1)
3708 000061D6 7513      <1> jnz     short a_u_pd_2
3709                  <1> ;
3710                  <1> ; empty PDE (it does not point to valid page table address)
3711 000061D8 E849F6FFFF <1> call    allocate_page ; (allocate a new page table)
3712 000061DD 7302      <1> jnc     short a_u_pd_1 ; OK... now, we have a new page table.
3713                  <1> ; cf = 1
3714                  <1> ; There is not a free memory page to allocate a new page table !!!
3715 000061DF 5E         <1> pop     esi ; *
3716                  <1> a_u_im_retn:
3717 000061E0 C3         <1> retn    ; return to 'sysalloc' with 'insufficient memory' error
3718                  <1> ;
3719                  <1> a_u_pd_1: ; clear the new page table content
3720                  <1> ; EAX = Physical (base) address of the new page table
3721 000061E1 E8B1F6FFFF <1> call    clear_page ; Clear page content
3722                  <1> ;
3723 000061E6 0C07      <1> or      al, PDE_A_PRESENT + PDE_A_WRITE + PDE_A_USER
3724                  <1> ; set bit 0, bit 1 and bit 2 to 1
3725                  <1> ; (present, writable, user)
3726 000061E8 8946FC    <1> mov     [esi-4], eax
3727                  <1> a_u_pd_2:
3728 000061EB 662500F0   <1> and     ax, PDE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3729                  <1> ; EAX = PHYSICAL (flat) ADDRESS OF THE PAGE TABLE
3730 000061EF 8B3C24    <1> mov     edi, [esp] ; *
3731                  <1> ; EDI = PTE index (of page directory)
3732                  <1> ; and edi, PTE_MASK ; PTE entry index in the page table
3733                  <1> ; EBX = next PDE address
3734 000061F2 89FE      <1> mov     esi, edi ; PTE index in page table (0-1023)
3735 000061F4 C1E702    <1> shl     edi, 2 ; convert PTE index to dword offset
3736 000061F7 01C7      <1> add     edi, eax ; now, edi points to requested PTE
3737                  <1> a_u_pt_0:
3738                  <1> ; 02/05/2017
3739 000061F9 8B07      <1> mov     eax, [edi]
3740                  <1> ;
3741 000061FB A801      <1> test    al, PTE_A_PRESENT ; bit 0, present flag (must be 1)
3742 000061FD 7445      <1> jz      short a_u_pt_1
3743                  <1> ;
3744 000061FF A802      <1> test    al, PTE_A_WRITE ; bit 1, writable (r/w) flag
3745                  <1> ; (must be 1)
3746 00006201 7550      <1> jnz     short a_u_pt_3
3747                  <1> ; Read only -duplicated- page (belongs to a parent or a child)
3748 00006203 66A90002   <1> test    ax, PTE_DUPLICATED ; was this page duplicated
3749                  <1> ; as child's page ?
3750 00006207 7455      <1> jz      short a_u_pt_4 ; Clear PTE but don't deallocate the page!
3751                  <1> ;
3752                  <1> ; check the parent's PTE value is read only & same page or not..
3753                  <1> ; EDX = page directory entry index (0-1023)
3754 00006209 52         <1> push    edx ; **
3755 0000620A 53         <1> push    ebx ; ***
3756                  <1> ; ESI = page table entry index (0-1023)
3757                  <1> esi ; **** ; 20/05/2017
3758 0000620B 8B1D[B08E0100] <1> mov     ebx, [u.pgpgdir] ; page directory of the parent process
3759 00006211 66C1E202   <1> shl     dx, 2 ; *4
3760 00006215 01D3      <1> add     ebx, edx ; PTE address, 0 (for the parent)
3761 00006217 8B13      <1> mov     edx, [ebx] ; page table address
3762 00006219 F6C201    <1> test    dl, PDE_A_PRESENT ; present (valid) or not ?
3763 0000621C 7433      <1> jz      short a_u_pt_2 ; parent process does not use this page
3764 0000621E 6681E200F0 <1> and     dx, PDE_A_CLEAR ; 0F000h ; Clear attribute bits
3765 00006223 66C1E602   <1> shl     si, 2 ; *4
3766                  <1> ; ESI = page table entry offset (0-4092)
3767 00006227 01D6      <1> add     esi, edx ; PTE address (for the parent)
3768 00006229 8B1E      <1> mov     ebx, [esi]
3769 0000622B F6C301    <1> test    bl, PTE_A_PRESENT ; present or not ?
3770 0000622E 7421      <1> jz      short a_u_pt_2 ; parent process does not use this page
3771 00006230 662500F0   <1> and     ax, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3772 00006234 6681E300F0 <1> and     bx, PTE_A_CLEAR ; 0F000h ; Clear attribute bits
3773 00006239 39D8      <1> cmp     eax, ebx ; parent's and child's pages are same ?
3774 0000623B 7514      <1> jne     short a_u_pt_2 ; not same page
3775                  <1> ; deallocate the child's page
3776 0000623D 800E02    <1> or      byte [esi], PTE_A_WRITE ; convert to writable page (parent)
3777                  <1> esi ; **** ; 20/05/2017
3778 00006240 5B         <1> pop     ebx ; ***
3779 00006241 5A         <1> pop     edx ; **
3780 00006242 EB1A      <1> jmp     short a_u_pt_4
3781                  <1> a_u_pt_1:
3782 00006244 09C0      <1> or      eax, eax ; swapped page ?
3783 00006246 7416      <1> jz      short a_u_pt_4 ; no
3784                  <1> ; yes
3785 00006248 D1E8      <1> shr     eax, 1
3786 0000624A E8B5F9FFFF <1> call    unlink_swap_block ; Deallocate swapped page block
3787                  <1> ; on the swap disk (or in file)
3788 0000624F EB0D      <1> jmp     short a_u_pt_4
3789                  <1> a_u_pt_2:
3790                  <1> esi ; **** ; 20/05/2017
3791 00006251 5B         <1> pop     ebx ; ***
3792 00006252 5A         <1> pop     edx ; **
3793                  <1> a_u_pt_3:
3794 00006253 66A90004   <1> test    ax, PTE_SHARED ; shared or direct memory access indicator
3795 00006257 7505      <1> jnz     short a_u_pt_4 ; AVL bit 1 = 1, do not deallocate this page!
3796                  <1> ;
3797                  <1> ; and ax, PTE_A_CLEAR ; 0F000h ; clear lower 12 (attribute) bits
3798 00006259 E887F7FFFF <1> call    deallocate_page ; set the mem allocation bit of this page
3799                  <1> ;
3800                  <1> a_u_pt_4:
3801 0000625E 89E8      <1> mov     eax, ebp ; physical address
3802 00006260 0C07      <1> or      al, PTE_A_PRESENT + PTE_A_WRITE + PTE_A_USER ; 04/03/2017
3803 00006262 AB         <1> stosd
3804 00006263 5E         <1> pop     esi ; * ; 20/05/2017
3805 00006264 49         <1> dec     ecx ; remain page count
3806 00006265 7417      <1> jz      short a_u_pd_5
3807 00006267 81C500100000 <1> add     ebp, PAGE_SIZE
3808 0000626D 46         <1> inc     esi ; next PTE (index)
3809                  <1> ; 20/05/2017
3810                  <1> ; cmp esi, PAGE_SIZE/4 ; 1024
3811                  <1> ; jnb short a_u_pt_0
3812 0000626E 6681E6FF03 <1> and     si, PTE_MASK ; 3FFh (0 to 1023)
3813 00006273 56         <1> push    esi ; *
3814 00006274 7583      <1> jnz     short a_u_pt_0 ; > 0 (<1024)
3815                  <1> a_u_pd_3:
3816                  <1> inc     edx
3817                  <1> ; cmp edx, 1024
3818                  <1> ; jnb short a_u_pd_4 ; 02/05/2017 (error!, ecx > 0)
3819 00006277 89DE      <1> mov     esi, ebx ; the next PDE address
3820 00006279 E953FFFFFF <1> jmp     a_u_pd_0
3821                  <1> a_u_pd_4:
3822                  <1> ; 02/05/2017
3823                  <1> ; stc
3824                  <1> a_u_pd_5:
3825                  <1> ; 20/05/2017
3826                  <1> ; pop edi ; *
3827 0000627E C3         <1> retn
3828                  <1> ;
3829                  <1> ; 28/11/2023
3830                  <1> ; a_lfb_k_err:

```



```

3831         <1>         ;retn
3832         <1>
3833         <1> allocate_lfb_pages_for_kernel:
3834         <1>         ; 02/12/2023
3835         <1>         ; 29/11/2023
3836         <1>         ; 28/11/2023
3837         <1>         ; 20/10/2023 - TRDOS 386 v2.0.7
3838         <1>         ; (only the overlapped main memory pages will be allocated)
3839         <1>         ; 28/11/2023
3840         <1>         ; (but all LFB pages will be added to the kernel's page tables)
3841         <1>         ; (NOTE: LFB size -which will be used- is the size of the max.
3842         <1>         ; recognized screen resolution. For example: 1024*768*4 bytes)
3843         <1>         ; 15/12/2020
3844         <1>         ; 14/12/2020 - TRDOS 386 v2.0.3
3845         <1>         ; Set kernel page tables for linear frame buffer
3846         <1>
3847         <1>         ; Input:
3848         <1>         ;         ; [LFB_ADDR] = linear frame buffer base address
3849         <1>         ;         ; [LFB_SIZE] = linear frame buffer size in bytes
3850         <1>         ;         ; 29/11/2023
3851         <1>         ;         ; ebx = Linear frame buffer base address as page num
3852         <1>         ;         ; 20/10/2023
3853         <1>         ;         ; esi = linear frame buffer base address
3854         <1>         ;         ; ecx = memory size in pages = [memory_size]
3855         <1>         ;         ; 28/11/2023
3856         <1>         ;         ; edx = linear frame buffer size in pages
3857         <1>         ; Output:
3858         <1>         ;         ; none
3859         <1>         ;         ; cf = 1 -> error
3860         <1>
3861         <1>         ; Modified registers: eax, ebx, ecx, edx, esi, edi, ebp
3862         <1>
3863         <1>
3864         <1> ; 02/12/2023 ; (!**!)
3865         <1> ; ('memory size' -for M.A.T.- is always <= LFB stat/base address)
3866         <1> %if 0
3867         <1>
3868         <1>         ; 29/11/2023
3869         <1>         ; allocate LFB blocks which < memory size at first
3870         <1>
3871         <1>         ;mov     eax, edx
3872         <1>         ;shl     eax, 12
3873         <1>         ;add     eax, esi
3874         <1>         ;jc      short a_lfb_k__err ; > 4GB limit
3875         <1>
3876         <1>         ;;mov     eax, 1048576 ; page count of (full) 4GB memory
3877         <1>         ;;sub     eax, edx
3878         <1>         ;;cmp     eax, esi
3879         <1>         ;jb      short a_lfb_k__err ; > 4GB limit
3880         <1>
3881         <1>         mov     ebx, esi ; LFB base/start address
3882         <1>         shr     ebx, 12 ; convert byte address to page address
3883         <1>         ; ebx = LFB start/base page number
3884         <1>
3885         <1>         sub     ecx, ebx
3886         <1>         jna     short a_lfb_k_2 ; skip M.A.T. bit allocation
3887         <1>         cmp     ecx, edx
3888         <1>         jna     short a_lfb_k_0
3889         <1>         mov     ecx, edx
3890         <1> a_lfb_k_0:
3891         <1>         sub     edx, ecx
3892         <1>         jna     short a_lfb_k_1
3893         <1>
3894         <1>         ; edx = remain count of LFB pages which are out of the M.A.T.
3895         <1>         ; ebx = LFB start/base page number
3896         <1>         ; ecx = page allocation count in the M.A.T.
3897         <1>
3898         <1>         ;mov     eax, 4096
3899         <1>         ;mul     ecx
3900         <1>         mov     eax, ecx
3901         <1>         shl     eax, 12 ; * 4096
3902         <1>
3903         <1>         add     esi, eax
3904         <1>         ; esi = LFB address just after the main memory if there is
3905         <1> a_lfb_k_1:
3906         <1>         mov     eax, ebx ; LFB start/base page number
3907         <1>         call    allocate_lfb_memory_block
3908         <1>
3909         <1>         ; here if there is, remain (the 2nd) part of the LFB
3910         <1>         ; will be allocated by using new kernel page table(s)
3911         <1>         ; (but M.A.T. bits will not be cleared because
3912         <1>         ; the 2nd LFB part is out of the M.A.T.)
3913         <1>
3914         <1>         and     edx, edx
3915         <1>         jz      short a_lfb_k_6 ; all LFB pages have been allocated
3916         <1>
3917         <1> %endif
3918         <1>
3919         <1> a_lfb_k_2:
3920         <1>         ; 29/11/2023
3921         <1>         shr     esi, 12 ; convert LFB address to page number
3922         <1>         ; edx = count of pages to be added (in kernel page tables)
3923         <1>         ; 28/11/2023
3924         <1>         mov     ebp, edx
3925         <1>         add     edx, 1023 ; round up
3926         <1>         shr     edx, 10 ; / 1024
3927         <1>         ; edx = count of kernel page tables to be added
3928         <1>         ; (edx = 1 or edx = 2)
3929         <1>         ; ((1024*768*4 = 3145728 bytes, 768 PTEs, 1 page table))
3930         <1>         ; ((1920*1080*4 = 8294400 bytes, 2025 PTEs, 2 page tables))
3931         <1>         call    allocate_page
3932         <1>         jc      short a_lfb_k__err
3933         <1>         ; (mem alloc error is not expected at this startup stage)
3934         <1>         ; eax = physical address of the new page table
3935         <1>         mov     edi, esi ; LFB start/base page number
3936         <1>         shr     edi, 10
3937         <1>         ; edi = PDE entry number of the Linear Frame Buffer addr
3938         <1>         ; (edi = 832 for 0D0000000h, edi = 896 for 0E0000000h)
3939         <1>         shl     edi, 2
3940         <1>         ; edi = PDE offset
3941         <1>         add     edi, [k_page_dir] ; Kernel's Page Dir Address
3942         <1>         push    eax ; +
3943         <1>         or      ax, PDE_A_PRESENT + PDE_A_WRITE + PDE_EXTERNAL
3944         <1>         ; supervisor + read&write + present
3945         <1>         ; + external memory block (LFB)
3946         <1>         stosd
3947         <1>
3948         <1>         dec     edx
3949         <1>         jz      short a_lfb_k_3 ; only 1 new page table
3950         <1>
3951         <1>         ; the 2nd page table
3952         <1>         call    allocate_page
3953         <1>         jc      short a_lfb_k__err
3954         <1>         ; (mem alloc error is not expected at this startup stage)

```

```

3955      <1>      ; eax = physical address of the new page table
3956 000062AE 89C3      <1>      mov     ebx, eax
3957 000062B0 660D0304  <1>      or      ax, PDE_A_PRESENT + PDE_A_WRITE + PDE_EXTERNAL
3958      <1>      ; supervisor + read&write + present
3959      <1>      ; + external memory block (LFB)
3960 000062B4 AB        <1>      stosd
3961      <1>
3962      <1>      a_lfb_k_3:
3963      <1>      ; set new PTEs
3964      <1>      ;
3965      <1>      ; ebp = count of pages (PTEs) to be allocated
3966 000062B5 5F        <1>      pop     edi ; + ; 1st page table address
3967      <1>      ; 29/11/2023
3968 000062B6 B900040000 <1>      mov     ecx, 1024 ; number of PTEs in a page table
3969 000062BB 89F0      <1>      mov     eax, esi ; LFB base/start page number
3970      <1>      ; (may be the 2nd part of the LFB)
3971 000062BD C1E00C      <1>      shl     eax, 12
3972 000062C0 660D0304  <1>      or      ax, PTE_A_PRESENT + PTE_A_WRITE + PTE_EXTERNAL
3973      <1>      ; supervisor + read&write + present
3974      <1>      ; + external memory block (LFB)
3975      <1>      a_lfb_k_4:
3976 000062C4 AB        <1>      stosd ; save as PTE
3977 000062C5 4D        <1>      dec     ebp
3978 000062C6 740D      <1>      jz      short a_lfb_k_5
3979 000062C8 0500100000 <1>      add     eax, 4096 ; the next page address
3980 000062CD E2F5      <1>      loop    a_lfb_k_4
3981      <1>
3982      <1>      ;and     ebx, ebx
3983      <1>      ;jz      short a_lfb_k_6
3984 000062CF 89DF      <1>      mov     edi, ebx ; the 2nd page table address
3985      <1>      ;xor     ebx, ebx
3986      <1>      ;mov     ecx, 1024
3987 000062D1 B504      <1>      ch, 4 ; cx = 4*256 = 1024
3988 000062D3 EBEF      <1>      jmp     short a_lfb_k_4 ; only 1 more loop (no the 3rd pt)
3989      <1>      ; (ebp <= 1024 here)
3990      <1>      a_lfb_k_5:
3991 000062D5 49        <1>      dec     ecx
3992 000062D6 7404      <1>      jz      short a_lfb_k_6
3993 000062D8 31C0      <1>      xor     eax, eax ; clear page table entry (empty)
3994 000062DA F3AB      <1>      rep     stosd
3995      <1>      a_lfb_k_6:
3996      <1>      ; 29/11/2023
3997 000062DC C3        <1>      retn
3998      <1>
3999      <1>      ; 28/11/2023
4000      <1>      %if 0
4001      <1>
4002      <1>      allocate_lfb_pages_for_kernel:
4003      <1>      ; 20/10/2023 - TRDOS 386 v2.0.7
4004      <1>      ; (only the overlapped main memory pages will be allocated)
4005      <1>      ; ((if LFB base/start address is greater than -or equal to-
4006      <1>      ; main memory size, this procedure is not called/used))
4007      <1>      ; 15/12/2020
4008      <1>      ; 14/12/2020 - TRDOS 386 v2.0.3
4009      <1>      ; Set kernel page tables for linear frame buffer
4010      <1>      ;
4011      <1>      ; Input:
4012      <1>      ; [LFB_ADDR] = linear frame buffer base address
4013      <1>      ; [LFB_SIZE] = linear frame buffer size in bytes
4014      <1>      ; 20/10/2023
4015      <1>      ; eax = Linear frame buffer base address as page num
4016      <1>      ; (eax < memory size)
4017      <1>      ; esi = linear frame buffer base address
4018      <1>      ; ecx = memory size in pages = [memory_size]
4019      <1>      ; Output:
4020      <1>      ; none
4021      <1>      ; cf = 1 -> error
4022      <1>      ;
4023      <1>      ; Modified registers: eax, ecx, edx, edi
4024      <1>
4025      <1>      ; 20/10/2023
4026      <1>      ;mov     edi, [LFB_ADDR]
4027      <1>      ;mov     edx, [LFB_SIZE]
4028      <1>
4029      <1>      ;;;
4030      <1>      ; 20/10/2023
4031      <1>      mov     edi, eax ; LFB base/start page number
4032      <1>      ;mov     ecx, [memory_size]
4033      <1>      sub     ecx, eax
4034      <1>
4035      <1>      ; ecx = number of pages to be set as (already) allocated
4036      <1>      ;;;
4037      <1>
4038      <1>      ; 20/10/2023
4039      <1>      shr     edi, 10 ; convert page number to PDE entry number
4040      <1>      ; (1024 pages per page table -the shift
4041      <1>      ; result is a page table index number
4042      <1>      ; or page directory entry number <= 1023)
4043      <1>      ;shr     edi, 22 ; convert address to page number
4044      <1>      ; and then convert it to PDE entry offset
4045      <1>      ; (1 PDE is for 4MB, 22 bit shift)
4046      <1>
4047      <1>      shl     di, 2 ; * 4 to obtain the PDE offset
4048      <1>
4049      <1>      ; 20/10/2023
4050      <1>      ;add     edx, 4095
4051      <1>      ;shr     edx, 12 ; convert LFB size to LFB page count
4052      <1>
4053      <1>      ;mov     ecx, edx ; * ; LFB page count
4054      <1>      ; 20/10/2023
4055      <1>      mov     edx, ecx ; *
4056      <1>      ; 20/10/2023
4057      <1>      ; edx = page count (to be set as allocated)
4058      <1>      ; from the LFB start page
4059      <1>      ; (it may be <= linear frame buffer size)
4060      <1>
4061      <1>      add     ecx, 1023 ; page count + 1023
4062      <1>      shr     ecx, 10 ; convert to page directory entry count
4063      <1>      ; (page table count)
4064      <1>      push    ecx ; **
4065      <1>      shl     ecx, 12 ; convert to byte count
4066      <1>
4067      <1>      ; 20/10/2023
4068      <1>      ; (absolute allocation is needed instead of the first fit)
4069      <1>      ;xor     eax, eax ; first available pages
4070      <1>
4071      <1>      ; 20/10/2023
4072      <1>      ;mov     esi, [LFB_ADDR]
4073      <1>      mov     eax, esi ; linear frame buffer's base/start address
4074      <1>
4075      <1>      ; allocate contiguous memory block for these kernel pages
4076      <1>
4077      <1>      ;call    allocate_memory_block
4078      <1>      ; 20/10/2023

```

```

4079      <1>      ; (short way to set LFB page bits on the 'M.A.T.')
4080      <1>      call    allocate_lfb_memory_block ; in 'allocate_memory_block'
4081      <1>
4082      <1>      ; eax = start address of (contiguous) memory block
4083      <1>      pop     ecx ; ** ; PDE count
4084      <1>      jnc     short a_lfb_k_1
4085      <1>      ; error (cf=1)
4086      <1>      retn
4087      <1> a_lfb_k_1:
4088      <1>      ; Allocate (new) page tables in kernel's page directory
4089      <1>      push    ecx ; PDE (page table) count
4090      <1>      push    eax ; start address of contiguous memory pages
4091      <1>      ; (at page boundary)
4092      <1>      ; edi = 1st page directory entry offset
4093      <1>      add     edi, [k_page_dir] ; Kernel's Page Dir Address
4094      <1> a_lfb_k_2:
4095      <1>      or      ax, PDE_A_PRESENT + PDE_A_WRITE + PDE_EXTERNAL
4096      <1>      ; supervisor + read&write + present
4097      <1>      ; + external memory block (LFB)
4098      <1>      stosd
4099      <1>      add     eax, 4096
4100      <1>      loop    a_lfb_k_2
4101      <1>
4102      <1>      pop     edi ; start addr of contiguous memory pages
4103      <1>      pop     ecx ; page table (PDE) count
4104      <1>
4105      <1>      ; Allocate pages in (new) kernel page tables
4106      <1>
4107      <1>      ; (Note: page tables are contiguous in physical memory)
4108      <1>      shl     ecx, 10 ; * 1024, convert to (total) PTE count
4109      <1>
4110      <1>      mov     eax, [LFB_ADDR]
4111      <1>      ; edx = LFB page count
4112      <1>      and     ax, ~4095 ; lw of LFB address is 0
4113      <1> a_lfb_k_3:
4114      <1>      or      ax, PTE_A_PRESENT + PTE_A_WRITE + PTE_EXTERNAL
4115      <1>      ; supervisor + read&write + present
4116      <1>      ; + external memory block (LFB)
4117      <1>      stosd
4118      <1>      dec     edx
4119      <1>      jz      short a_lfb_k_4 ; LFB size has been completed (!?)
4120      <1>      add     eax, 4096
4121      <1>      loop    a_lfb_k_3
4122      <1>
4123      <1>      retn
4124      <1>
4125      <1> a_lfb_k_4:
4126      <1>      ; clear PTEs for empty/free pages
4127      <1>      ; (if there are after LFB !?)
4128      <1>      xor     eax, eax ; clear page table entry (empty)
4129      <1>      rep     stosd
4130      <1>      retn
4131      <1>
4132      <1> %endif
4133      <1>
4134      <1> ;deallocate_lfb_pages_for_kernel:
4135      <1>      ; 15/12/2020
4136      <1>      ; 14/12/2020 - TRDOS 386 v2.0.3
4137      <1>      ; Reset/Release kernel page tables
4138      <1>      ; which are used for linear frame buffer
4139      <1>      ; (this procedure will be called by kernel only)
4140      <1>      ;
4141      <1>      ; Input:
4142      <1>      ; [LFB_ADDR] = linear frame buffer base address
4143      <1>      ; [LFB_SIZE] = linear frame buffer size in bytes
4144      <1>      ; Output:
4145      <1>      ; none
4146      <1>      ;
4147      <1>      ; Modified registers: eax, ecx, edi
4148      <1>
4149      <1>      mov     edi, [LFB_ADDR]
4150      <1>      mov     ecx, [LFB_SIZE]
4151      <1>
4152      <1>      shr     edi, 22 ; convert address to page number
4153      <1>      ; and then convert it to PDE entry offset
4154      <1>      ; (1 PDE is for 4MB, 22 bit shift)
4155      <1>
4156      <1>      shl     di, 2 ; * 4 for offset
4157      <1>
4158      <1>      add     ecx, 4095
4159      <1>      shr     ecx, 12 ; convert LFB size to page count
4160      <1>
4161      <1>      add     ecx, 1023 ; page count + 1023
4162      <1>      shr     ecx, 10 ; convert to page directory entry count
4163      <1>      ; (page table count)
4164      <1>      push    ecx ; *
4165      <1>      shl     ecx, 12 ; convert to byte count
4166      <1>
4167      <1>      xor     eax, eax ; first available pages
4168      <1>
4169      <1>      ; deallocate contiguous memory block for kernel pages
4170      <1>
4171      <1>      call    deallocate_memory_block
4172      <1>
4173      <1>      pop     ecx ; * ; PDE count
4174      <1>
4175      <1>      ; Release/Free PDEs (page tables) in kernel's page dir
4176      <1>      ; edi = 1st page directory entry offset
4177      <1>      add     edi, [k_page_dir] ; Kernel's Page Dir Address
4178      <1>      sub     eax, eax ; clear (also invalidate)
4179      <1>      rep     stosd
4180      <1>
4181      <1>      retn
4182      <1>
4183      <1> ; /// End Of MEMORY MANAGEMENT FUNCTIONS ///
4184      <1>
4185      <1> ;; Data:
4186      <1>
4187      <1> ; 09/03/2015
4188      <1> ;swpq_count: dw 0 ; count of pages on the swap que
4189      <1> ;swpd_drv: dd 0 ; logical drive description table address of the swap drive/disk
4190      <1> ;swpd_size: dd 0 ; size of swap drive/disk (volume) in sectors (512 bytes).
4191      <1> ;swpd_free: dd 0 ; free page blocks (4096 bytes) on swap disk/drive (logical)
4192      <1> ;swpd_next: dd 0 ; next free page block
4193      <1> ;swpd_last: dd 0 ; last swap page block
4194      <1> %include 'timer.s' ; 17/01/2015
4195      <1> ; *****
4196      <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.5 - timer.s
4197      <1> ;
4198      <1> ; -----
4199      <1> ; Last Update: 08/08/2022 (Previous: 18/04/2021)
4200      <1> ; -----
4201      <1> ; Beginning: 17/01/2016
4202      <1> ; -----
4203      <1> ; Assembler: NASM version 2.15 (trdos386.s)

```

```

9      <1> ; -----
10     <1> ; Turkish Rational DOS
11     <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
12     <1> ;
13     <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
14     <1> ;
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> ;
25     <1> ; //////////// TIMER (& REAL TIME CLOCK) FUNCTIONS ////////////
26     <1> ;
27     <1> int1Ah:
28     <1> ; 29/01/2016
29     <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
30     <1> pushfd
31 000062DD 9C     <1> push cs
32 000062DE 0E     <1> call TIME_OF_DAY_1
33 000062E4 C3     <1> retn
34     <1> ;
35     <1> ; -----
36     <1> ;
37     <1> ; --- INT 1A H -- (TIME OF DAY) -----
38     <1> ; THIS BIOS ROUTINE ALLOWS THE CLOCKS TO BE SET OR READ
39     <1> ;
40     <1> ; PARAMETERS:
41     <1> ; (AH) = 00H READ THE CURRENT SETTING AND RETURN WITH,
42     <1> ; (CX) = HIGH PORTION OF COUNT
43     <1> ; (DX) = LOW PORTION OF COUNT
44     <1> ; (AL) = 0 TIMER HAS NOT PASSED 24 HOURS SINCE LAST READ
45     <1> ; 1 IF ON ANOTHER DAY. (RESET TO ZERO AFTER READ)
46     <1> ;
47     <1> ; (AH) = 01H SET THE CURRENT CLOCK USING,
48     <1> ; (CX) = HIGH PORTION OF COUNT
49     <1> ; (DX) = LOW PORTION OF COUNT.
50     <1> ;
51     <1> ; NOTE: COUNTS OCCUR AT THE RATE OF 1193180/65536 COUNTS/SECOND
52     <1> ; (OR ABOUT 18.2 PER SECOND -- SEE EQUATES)
53     <1> ;
54     <1> ; (AH) = 02H READ THE REAL TIME CLOCK AND RETURN WITH,
55     <1> ; (CH) = HOURS IN BCD (00-23)
56     <1> ; (CL) = MINUTES IN BCD (00-59)
57     <1> ; (DH) = SECONDS IN BCD (00-59)
58     <1> ; (DL) = DAYLIGHT SAVINGS ENABLE (00-01)
59     <1> ;
60     <1> ; (AH) = 03H SET THE REAL TIME CLOCK USING,
61     <1> ; (CH) = HOURS IN BCD (00-23)
62     <1> ; (CL) = MINUTES IN BCD (00-59)
63     <1> ; (DH) = SECONDS IN BCD (00-59)
64     <1> ; (DL) = 01 IF DAYLIGHT SAVINGS ENABLE OPTION, ELSE 00.
65     <1> ;
66     <1> ; NOTE: (DL) = 00 IF DAYLIGHT SAVINGS TIME ENABLE IS NOT ENABLED.
67     <1> ; (DL) = 01 ENABLES TWO SPECIAL UPDATES THE LAST SUNDAY IN
68     <1> ; APRIL (1:59:59 --> 3:00:00 AM) AND THE LAST SUNDAY IN
69     <1> ; OCTOBER (1:59:59 --> 1:00:00 AM) THE FIRST TIME.
70     <1> ;
71     <1> ; (AH) = 04H READ THE DATE FROM THE REAL TIME CLOCK AND RETURN WITH,
72     <1> ; (CH) = CENTURY IN BCD (19 OR 20)
73     <1> ; (CL) = YEAR IN BCD (00-99)
74     <1> ; (DH) = MONTH IN BCD (01-12)
75     <1> ; (DL) = DAY IN BCD (01-31).
76     <1> ;
77     <1> ; (AH) = 05H SET THE DATE INTO THE REAL TIME CLOCK USING,
78     <1> ; (CH) = CENTURY IN BCD (19 OR 20)
79     <1> ; (CL) = YEAR IN BCD (00-99)
80     <1> ; (DH) = MONTH IN BCD (01-12)
81     <1> ; (DL) = DAY IN BCD (01-31).
82     <1> ;
83     <1> ; (AH) = 06H SET THE ALARM TO INTERRUPT AT SPECIFIED TIME,
84     <1> ; (CH) = HOURS IN BCD (00-23 (OR FFH))
85     <1> ; (CL) = MINUTES IN BCD (00-59 (OR FFH))
86     <1> ; (DH) = SECONDS IN BCD (00-59 (OR FFH))
87     <1> ;
88     <1> ; (AH) = 07H RESET THE ALARM INTERRUPT FUNCTION.
89     <1> ;
90     <1> ; NOTES: FOR ALL RETURNS CY= 0 FOR SUCCESSFUL OPERATION.
91     <1> ; FOR (AH)= 2, 4, 6 - CARRY FLAG SET IF REAL TIME CLOCK NOT OPERATING.
92     <1> ; FOR (AH)= 6 - CARRY FLAG SET IF ALARM ALREADY ENABLED.
93     <1> ; FOR THE ALARM FUNCTION (AH = 6) THE USER MUST SUPPLY A ROUTINE AND
94     <1> ; INTERCEPT THE CORRECT ADDRESS IN THE VECTOR TABLE FOR INTERRUPT 4AH.
95     <1> ; USE 0FFH FOR ANY "DO NOT CARE" POSITION FOR INTERVAL INTERRUPTS.
96     <1> ; INTERRUPTS ARE DISABLED DURING DATA MODIFICATION.
97     <1> ; AH & AL ARE RETURNED MODIFIED AND NOT DEFINED EXCEPT WHERE INDICATED.
98     <1> ; -----
99     <1> ;
100    <1> ; 29/07/2022
101    <1> ; 15/01/2017
102    <1> ; 14/01/2017
103    <1> ; 07/01/2017
104    <1> ; 02/01/2017
105    <1> ; 29/05/2016
106    <1> ; 29/01/2016
107    <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
108    <1> ;
109    <1> ; 29/05/2016
110    <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
111    <1> int35h: ; Date/Time functions
112    <1> ;
113    <1> TIME_OF_DAY_1:
114    <1> ; 07/08/5022
115    <1> ; 29/07/2022 (TRDOS 386 Kernel v2.0.5)
116    <1> ; sti ; INTERRUPTS BACK ON
117    <1> ; 29/05/2016
118 000062E5 80642408FE <1> and byte [esp+8], 11111110b ; clear carry bit of eflags register
119    <1> ;
120 000062EA 80FC08 <1> cmp ah, (RTC_TBE-RTC_TB)/4; CHECK IF COMMAND IN VALID RANGE (0-7)
121 000062ED F5 <1> cmc ; COMPLEMENT CARRY FOR ERROR EXIT
122    <1> ; (*) jc short TIME_9 ; EXIT WITH CARRY = 1 IF NOT VALID
123 000062EE 721A <1> jc short _TIME_9 ; 29/05/2016
124    <1> ;
125 000062F0 1E <1> push ds
126 000062F1 56 <1> push esi
127 000062F2 66BE1000 <1> mov si, KDATA ; kernel data segment
128 000062F6 8EDE <1> mov ds, si
129    <1> ;
130    <1> ; 15/01/2017
131    <1> ; 14/01/2017
132    <1> ; 02/01/2017

```

```

133      <1>      ;;mov    byte [intflg], 35h      ; date & time interrupt
134      <1>      ;;sti
135      <1>      ;;
136      000062F8 C0E402      <1>      shl     ah, 2          ; convert function to dword offset
137      000062FB 0FB6F4      <1>      movzx   esi, ah        ; PLACE INTO ADDRESSING REGISTER
138      <1>      ;;cli
139      000062FE FF96[10630000] <1>      call    [esi+RTC_TB]      ; NO INTERRUPTS DURING TIME FUNCTIONS
140      <1>      ;;
141      <1>      ;;sti
142      00006304 B400      <1>      mov     ah, 0          ; RETURN WITH CARRY FLAG SET FOR RESULT
143      00006306 5E      <1>      pop     esi          ; INTERRUPTS BACK ON
144      00006307 1F      <1>      pop     ds          ; CLEAR (AH) TO ZERO
145      <1>      ;;
146      <1>      ;;15/01/2017
147      <1>      ;; 02/01/2017
148      <1>      ;;mov    byte [ss:intflg], 0 ; 07/01/2017
149      <1>      ;;
150      <1>      ;TIME_9:
151      <1>      ;;
152      <1>      ; (*) 29/05/2016          ; RETURN WITH CY= 0 IF NO ERROR
153      <1>      ; (*) retf 4 ; skip eflags on stack
154      00006308 7305      <1>      jnc     short _TIME_10
155      <1>      _TIME_9:
156      <1>      ; 29/05/2016 -set carry flag on stack-
157      <1>      ; [esp] = EIP
158      <1>      ; [esp+4] = CS
159      <1>      ; [esp+8] = E-FLAGS
160      0000630A 804C240801 <1>      or      byte [esp+8], 1      ; set carry bit of eflags register
161      <1>      ; [esp+12] = ESP (user)
162      <1>      ; [esp+16] = SS (User)
163      <1>      _TIME_10:
164      0000630F CF      <1>      iretd
165      <1>      ;;
166      <1>      ; (*) 29/05/2016 - 'ref 4' intruction causes to stack fault
167      <1>      ; (OUTER-PRIVILEGE-LEVEL)
168      <1>      ; INTEL 80386 PROGRAMMER'S REFERENCE MANUAL 1986
169      <1>      ; // RETF instruction:
170      <1>      ;;
171      <1>      ; IF OperandMode=32 THEN
172      <1>      ; Load CS:EIP from stack;
173      <1>      ; Set CS RPL to CPL;
174      <1>      ; Increment ESP by 8 plus the immediate offset if it exists;
175      <1>      ; Load SS:ESP from stack;
176      <1>      ; ELSE (* OperandMode=16 *)
177      <1>      ; Load CS:IP from stack;
178      <1>      ; Set CS RPL to CPL;
179      <1>      ; Increment ESP by 4 plus the immediate offset if it exists;
180      <1>      ; Load SS:ESP from stack;
181      <1>      ; FI;
182      <1>      ;;
183      <1>      ; //
184      <1>      ;;
185      <1>      ; ROUTINE VECTOR TABLE (AH)=
186      00006310 [30630000] <1>      RTC_TB:
187      00006314 [43630000] <1>      dd      RTC_00          ; 0 = READ CURRENT CLOCK COUNT
188      00006318 [51630000] <1>      dd      RTC_10          ; 1 = SET CLOCK COUNT
189      0000631C [7F630000] <1>      dd      RTC_20          ; 2 = READ THE REAL TIME CLOCK TIME
190      00006320 [BC630000] <1>      dd      RTC_30          ; 3 = SET REAL TIME CLOCK TIME
191      00006324 [E2630000] <1>      dd      RTC_40          ; 4 = READ THE REAL TIME CLOCK DATE
192      00006328 [41640000] <1>      dd      RTC_50          ; 5 = SET REAL TIME CLOCK DATE
193      0000632C [93640000] <1>      dd      RTC_60          ; 6 = SET THE REAL TIME CLOCK ALARM
194      <1>      dd      RTC_70          ; 7 = RESET ALARM
195      <1>      RTC_TBE equ      $
196      <1>      ;;
197      <1>      RTC_00:
198      00006330 A0[24770100] <1>      mov     al, [TIMER_OFL]      ; READ TIME COUNT
199      00006335 C605[24770100]00 <1>      mov     byte [TIMER_OFL], 0      ; GET THE OVERFLOW FLAG
200      0000633C 8B0D[20770100] <1>      mov     ecx, [TIMER_LH]      ; AND THEN RESET THE OVERFLOW FLAG
201      00006342 C3      <1>      retn          ; GET COUNT OF TIME
202      <1>      ;;
203      <1>      RTC_10:
204      00006343 890D[20770100] <1>      mov     [TIMER_LH], ecx      ; SET TIME COUNT
205      00006349 C605[24770100]00 <1>      mov     byte [TIMER_OFL], 0      ; SET TIME COUNT
206      00006350 C3      <1>      retn          ; RESET OVERFLOW FLAG
207      <1>      ;;
208      <1>      RTC_20:
209      00006351 E8C7010000 <1>      call    UPD_IPR          ; GET RTC TIME
210      00006356 7226      <1>      jc      short RTC_29      ; CHECK FOR UPDATE IN PROCESS
211      <1>      ;;
212      <1>      mov     al, CMOS_SECONDS      ; SET ADDRESS OF SECONDS
213      0000635A E8F4010000 <1>      call    CMOS_READ      ; GET SECONDS
214      0000635F 88C6      <1>      mov     dh, al          ; SAVE
215      00006361 B00B      <1>      mov     al, CMOS_REG_B      ; ADDRESS ALARM REGISTER
216      00006363 E8EB010000 <1>      call    CMOS_READ      ; READ CURRENT VALUE OF DSE BIT
217      00006368 2401      <1>      and     al, 00000001b      ; MASK FOR VALID DSE BIT
218      0000636A 88C2      <1>      mov     dl, al          ; SET [DL] TO ZERO FOR NO DSE BIT
219      0000636C B002      <1>      mov     al, CMOS_MINUTES      ; SET ADDRESS OF MINUTES
220      0000636E E8E0010000 <1>      call    CMOS_READ      ; GET MINUTES
221      00006373 88C1      <1>      mov     cl, al          ; SAVE
222      00006375 B004      <1>      mov     al, CMOS_HOURS      ; SET ADDRESS OF HOURS
223      <1>      RTC_41:
224      00006377 E8D7010000 <1>      call    CMOS_READ      ; GET HOURS
225      0000637C 88C5      <1>      mov     ch, al          ; SAVE
226      <1>      ; 29/07/2022
227      <1>      ;clc
228      <1>      RTC_29:
229      0000637E C3      <1>      retn          ; SET CY= 0
230      <1>      ;;
231      <1>      RTC_30:
232      0000637F E899010000 <1>      call    UPD_IPR          ; SET RTC TIME
233      00006384 7305      <1>      jnc     short RTC_35      ; CHECK FOR UPDATE IN PROCESS
234      00006386 E8AD010000 <1>      call    RTC_STA          ; GO AROUND IF CLOCK OPERATING
235      <1>      RTC_35:
236      0000638B 88F4      <1>      mov     ah, dh          ; GET TIME BYTE - SECONDS
237      0000638D B000      <1>      mov     al, CMOS_SECONDS      ; ADDRESS SECONDS
238      0000638F E894000000 <1>      call    CMOS_WRITE      ; UPDATE SECONDS
239      00006394 88CC      <1>      mov     ah, cl          ; GET TIME BYTE - MINUTES
240      00006396 B002      <1>      mov     al, CMOS_MINUTES      ; ADDRESS MINUTES
241      00006398 E88B000000 <1>      call    CMOS_WRITE      ; UPDATE MINUTES
242      0000639D 88EC      <1>      mov     ah, ch          ; GET TIME BYTE - HOURS
243      0000639F B004      <1>      mov     al, CMOS_HOURS      ; ADDRESS HOURS
244      000063A1 E882000000 <1>      call    CMOS_WRITE      ; UPDATE ADDRESS
245      <1>      ;mov     al, CMOS_REG_B      ; ADDRESS ALARM REGISTER
246      <1>      ;mov     ah, al
247      000063A6 66B80B0B <1>      mov     ax, CMOS_REG_B * 257
248      000063AA E8A4010000 <1>      call    CMOS_READ      ; READ CURRENT TIME
249      000063AF 2462      <1>      and     al, 01100010b      ; MASK FOR VALID BIT POSITIONS
250      000063B1 0C02      <1>      or      al, 00000010b      ; TURN ON 24 HOUR MODE
251      000063B3 80E201 <1>      and     dl, 00000001b      ; USE ONLY THE DSE BIT
252      000063B6 08D0      <1>      or      al, dl          ; GET DAY LIGHT SAVINGS TIME BIT (OSE)
253      000063B8 86C4      <1>      xchg    ah, al          ; PLACE IN WORK REGISTER AND GET ADDRESS
254      <1>      ;call    CMOS_WRITE
255      <1>      ;clc
256      <1>      ;retn          ; SET NEW ALARM SITS

```

```

257      ; 29/07/2022
258 000063BA EB6C      <1>      jmp      short CMOS_WRITE
259      <1>
260      <1> RTC_40:      ; GET RTC DATE
261 000063BC E85C010000 <1>      call     UPD_IPR      ; CHECK FOR UPDATE IN PROCESS
262      <1>      ;jc      short RTC_49      ; EXIT IF ERROR (CY= 1)
263      <1>      ; 07/08/2022
264 000063C1 72BB      <1>      jc      short RTC_29
265      <1>
266 000063C3 B007      <1>      mov     al, CMOS_DAY_MONTH      ; ADDRESS DAY OF MONTH
267 000063C5 E889010000 <1>      call     CMOS_READ      ; READ DAY OF MONTH
268 000063CA 88C2      <1>      mov     dl, al      ; SAVE
269 000063CC B008      <1>      mov     al, CMOS_MONTH      ; ADDRESS MONTH
270 000063CE E880010000 <1>      call     CMOS_READ      ; READ MONTH
271 000063D3 88C6      <1>      mov     dh, al      ; SAVE
272 000063D5 B009      <1>      mov     al, CMOS_YEAR      ; ADDRESS YEAR
273 000063D7 E877010000 <1>      call     CMOS_READ      ; READ YEAR
274 000063DC 88C1      <1>      mov     cl, al      ; SAVE
275 000063DE B032      <1>      mov     al, CMOS_CENTURY      ; ADDRESS CENTURY LOCATION
276      <1> ; 29/07/2022
277      <1> ;      call     CMOS_READ      ; GET CENTURY BYTE
278      <1> ;      mov     ch, al      ; SAVE
279      <1> ;      ; 29/07/2022
280      <1> ;      ;clic      ; SET CY=0
281      <1> ;RTC_49:
282      <1> ;      retn      ; RETURN WITH RESULTS IN CARRY FLAG
283      <1>
284      <1> ; 29/07/2022
285 000063E0 EB95      <1>      jmp      short RTC_41
286      <1>
287      <1>
288      <1> RTC_50:      ; SET RTC DATE
289 000063E2 E836010000 <1>      call     UPD_IPR      ; CHECK FOR UPDATE IN PROCESS
290 000063E7 7305      <1>      jnc      short RTC_55      ; GO AROUND IF NO ERROR
291 000063E9 E84A010000 <1>      call     RTC_STA      ; ELSE INITIALIZE CLOCK
292      <1> RTC_55:
293 000063EE 66B80600 <1>      mov     ax, CMOS_DAY_WEEK      ; ADDRESS OF DAY OF WEEK BYTE
294 000063F2 E831000000 <1>      call     CMOS_WRITE      ; LOAD ZEROS TO DAY OF WEEK
295 000063F7 88D4      <1>      mov     ah, dl      ; GET DAY OF MONTH BYTE
296 000063F9 B007      <1>      mov     al, CMOS_DAY_MONTH      ; ADDRESS DAY OF MONTH BYTE
297 000063FB E828000000 <1>      call     CMOS_WRITE      ; WRITE OF DAY OF MONTH REGISTER
298 00006400 88F4      <1>      mov     ah, dh      ; GET MONTH
299 00006402 B008      <1>      mov     al, CMOS_MONTH      ; ADDRESS MONTH BYTE
300 00006404 E81F000000 <1>      call     CMOS_WRITE      ; WRITE MONTH REGISTER
301 00006409 88CC      <1>      mov     ah, cl      ; GET YEAR BYTE
302 0000640B B009      <1>      mov     al, CMOS_YEAR      ; ADDRESS YEAR REGISTER
303 0000640D E816000000 <1>      call     CMOS_WRITE      ; WRITE YEAR REGISTER
304 00006412 88EC      <1>      mov     ah, ch      ; GET CENTURY BYTE
305 00006414 B032      <1>      mov     al, CMOS_CENTURY      ; ADDRESS CENTURY BYTE
306 00006416 E80D000000 <1>      call     CMOS_WRITE      ; WRITE CENTURY LOCATION
307      <1> ;mov     al, CMOS_REG_B      ; ADDRESS ALARM REGISTER
308      <1> ;mov     ah, al
309 0000641B 66B80B0B <1>      mov     ax, CMOS_REG_B * 257
310 0000641F E82F010000 <1>      call     CMOS_READ      ; READ CURRENT SETTINGS
311 00006424 247F      <1>      and     al, 07Fh      ; CLEAR 'SET BIT'
312 00006426 86C4      <1>      xchg     ah, al      ; MOVE TO WORK REGISTER
313      <1> ;call     CMOS_WRITE      ; AND START CLOCK UPDATING
314      <1> ;clic      ; SET CY= 0
315      <1> ;retn      ; RETURN CY=0
316      <1> ; 29/07/2022
317      <1> ;jmp      short CMOS_WRITE
318      <1>
319      <1> ;-----
320      <1>
321      <1> ; 08/08/2022
322      <1> ; 29/07/2022 (TRDOS 386 v2.0.5)
323      <1> ; 18/04/2021 (TRDOS 386 v2.0.4)
324      <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
325      <1>
326      <1> ;--- CMOS_WRITE -----
327      <1> ; WRITE BYTE TO CMOS SYSTEM CLOCK CONFIGURATION TABLE      :
328      <1> ;      :
329      <1> ; INPUT: (AL)=      CMOS TABLE ADDRESS TO BE WRITTEN TO      :
330      <1> ;      BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT      :
331      <1> ;      BITS 6-0 = ADDRESS OF TABLE LOCATION TO WRITE      :
332      <1> ;      (AH)= NEW VALUE TO BE PLACED IN THE ADDRESSED TABLE LOCATION      :
333      <1> ;      :
334      <1> ; OUTPUT: VALUE IN (AH) PLACED IN LOCATION (AL) WITH NMI LEFT DISABLED      :
335      <1> ;      IF BIT 7 OF (AL) IS ON, DURING THE CMOS UPDATE BOTH NMI AND      :
336      <1> ;      NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY.      :
337      <1> ;      THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND      :
338      <1> ;      THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN.      :
339      <1> ;      ONLY THE CMOS LOCATION AND THE NMI STATE IS CHANGED.      :
340      <1> ;-----
341      <1>
342      <1> ; 08/08/2022
343      <1> CMOS_WRITE:      ; WRITE (AH) TO LOCATION (AL)
344      <1> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
345      <1> ;pushf      ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
346      <1> ;push ax      ; SAVE WORK REGISTER VALUES
347      <1> ; 18/04/2021
348      <1> ;push     eax
349 00006428 D0C0      <1>      rol     al, 1      ; MOVE NMI BIT TO LOW POSITION
350 0000642A F9      <1>      stc      ; FORCE NMI BIT ON IN CARRY FLAG
351 0000642B D0D8      <1>      rcr     al, 1      ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
352 0000642D FA      <1>      cli      ; DISABLE INTERRUPTS
353 0000642E E670      <1>      out     CMOS_PORT, al      ; ADDRESS LOCATION AND DISABLE NMI
354 00006430 88E0      <1>      mov     al, ah      ; GET THE DATA BYTE TO WRITE
355 00006432 E671      <1>      out     CMOS_DATA, al      ; PLACE IN REQUESTED CMOS LOCATION
356 00006434 B01E      <1>      mov     al, CMOS_SHUT_DOWN*2      ; GET ADDRESS OF DEFAULT LOCATION
357      <1> ;mov     al, CMOS_REG_D*2      ; GET ADDRESS OF DEFAULT LOCATION
358 00006436 D0D8      <1>      rcr     al, 1      ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
359 00006438 E670      <1>      out     CMOS_PORT, al      ; SET DEFAULT TO READ ONLY REGISTER
360      <1> ;nop      ; I/O DELAY
361      <1> ; 29/07/2022
362 0000643A E6EB      <1>      out     0EBh, al ; NEWIODELAY ; AWARD BIOS 1999, ATIME.ASM
363 0000643C E471      <1>      in      al, CMOS_DATA      ; OPEN STANDBY LATCH
364      <1> ;pop     ax      ; RESTORE WORK REGISTERS
365      <1> ; 18/04/2021
366      <1> ;pop     eax
367      <1> ;popf
368      <1> ; 29/07/2022
369      <1> ;clic
370      <1> ; 08/08/2022
371 0000643E 30C0      <1>      xor     al, al
372 00006440 C3      <1>      retn
373      <1>
374      <1> ;-----
375      <1>
376      <1> RTC_60:      ; SET RTC ALARM
377 00006441 B00B      <1>      mov     al, CMOS_REG_B      ; ADDRESS ALARM
378 00006443 E80B010000 <1>      call     CMOS_READ      ; READ ALARM REGISTER
379 00006448 A820      <1>      test    al, 20h      ; CHECK FOR ALARM ALREADY ENABLED
380 0000644A F9      <1>      stc      ; SET CARRY IN CASE OF ERROR

```

```

381 0000644B 7541      <1>      jnz      short RTC_69      ; ERROR EXIT IF ALARM SET
382 0000644D E8CB000000 <1>      call     UPD_IPR      ; CHECK FOR UPDATE IN PROCESS
383 00006452 7305      <1>      jnc      short RTC_65      ; SKIP INITIALIZATION IF NO ERROR
384 00006454 E8DF000000 <1>      call     RTC_STA      ; ELSE INITIALIZE CLOCK
385                                     <1> RTC_65:
386 00006459 88F4      <1>      mov     ah, dh      ; GET SECONDS BYTE
387 0000645B B001      <1>      mov     al, CMOS_SEC_ALARM ; ADDRESS THE SECONDS ALARM REGISTER
388 0000645D E8C6FFFFFF <1>      call     CMOS_WRITE ; INSERT SECONDS
389 00006462 88CC      <1>      mov     ah, cl      ; GET MINUTES PARAMETER
390 00006464 B003      <1>      mov     al, CMOS_MIN_ALARM ; ADDRESS MINUTES ALARM REGISTER
391 00006466 E8BDFFFFFF <1>      call     CMOS_WRITE ; INSERT MINUTES
392 00006468 88EC      <1>      mov     ah, ch      ; GET HOURS PARAMETER
393 0000646D B005      <1>      mov     al, CMOS_HR_ALARM ; ADDRESS HOUR ALARM REGISTER
394 0000646F E8B4FFFFFF <1>      call     CMOS_WRITE ; INSERT HOURS
395 00006474 E4A1      <1>      in      al, INTB01 ; READ SECOND INTERRUPT MASK REGISTER
396 00006476 24FE      <1>      and     al, 0FEh ; ENABLE ALARM TIMER BIT (CY= 0)
397 00006478 E6A1      <1>      out     INTB01, al ; WRITE UPDATED MASK
398                                     <1>      ;mov     al, CMOS_REG_B ; ADDRESS ALARM REGISTER
399                                     <1>      ;mov     ah, al
400 0000647A 66B80B0B <1>      mov     ax, CMOS_REG_B * 257
401 0000647E E8D0000000 <1>      call     CMOS_READ ; READ CURRENT ALARM REGISTER
402 00006483 247F      <1>      and     al, 07Fh ; ENSURE SET BIT TURNED OFF
403 00006485 0C20      <1>      or      al, 20h ; TURN ON ALARM ENABLE
404 00006487 86C4      <1>      xchg    ah, al ; MOVE MASK TO OUTPUT REGISTER
405 00006489 E89AFFFFF <1>      call     CMOS_WRITE ; WRITE NEW ALARM MASK
406                                     <1>      ; 29/07/2022
407                                     <1>      ;clc
408                                     <1>      ; SET CY= 0
409 0000648E 66B80000 <1>      mov     ax, 0 ; CLEAR AX REGISTER
410 00006492 C3        <1>      retn    ; RETURN WITH RESULTS IN CARRY FLAG
411                                     <1>
412                                     <1> RTC_70: ; RESET ALARM
413                                     <1>      ;mov     al, CMOS_REG_B ; ADDRESS ALARM REGISTER
414                                     <1>      ;mov     ah, al
415 00006493 66B80B0B <1>      mov     ax, CMOS_REG_B * 257 ; ADDRESS ALARM REGISTER (TO BOTH AH,AL)
416 00006497 E8B7000000 <1>      call     CMOS_READ ; READ ALARM REGISTER
417 0000649C 2457      <1>      and     al, 57h ; TURN OFF ALARM ENABLE
418 0000649E 86C4      <1>      xchg    ah, al ; SAVE DATA AND RECOVER ADDRESS
419                                     <1>      ;call     CMOS_WRITE ; RESTORE NEW VALUE
420                                     <1>      ;clc ; SET CY= 0
421                                     <1>      ;retn ; RETURN WITH NO CARRY
422                                     <1>      ; 29/07/2022
423 000064A0 EB86      <1>      jmp     short CMOS_WRITE
424                                     <1>
425                                     <1> ;-----
426                                     <1>
427                                     <1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
428                                     <1>
429                                     <1> ;--- HARDWARE INT 70 H -- ( IRQ LEVEL 8) -----
430                                     <1> ; ALARM INTERRUPT HANDLER (RTC) :
431                                     <1> ; THIS ROUTINE HANDLES THE PERIODIC AND ALARM INTERRUPTS FROM THE CMOS :
432                                     <1> ; TIMER. INPUT FREQUENCY IS 1.024 KHZ OR APPROXIMATELY 1024 INTERRUPTS :
433                                     <1> ; EVERY SECOND FOR THE PERIODIC INTERRUPT. FOR THE ALARM FUNCTION, :
434                                     <1> ; THE INTERRUPT WILL OCCUR AT THE DESIGNATED TIME. :
435                                     <1> ; :
436                                     <1> ; INTERRUPTS ARE ENABLED WHEN THE EVENT OR ALARM FUNCTION IS ACTIVATED. :
437                                     <1> ; FOR THE EVENT INTERRUPT, THE HANDLER WILL DECREMENT THE WAIT COUNTER :
438                                     <1> ; AND WHEN IT EXPIRES WILL SET THE DESIGNATED LOCATION TO 80H. FOR :
439                                     <1> ; THE ALARM INTERRUPT. THE USER MUST PROVIDE A ROUTINE TO INTERCEPT :
440                                     <1> ; THE CORRECT ADDRESS FROM THE VECTOR TABLE INVOKED BY INTERRUPT 4AH :
441                                     <1> ; PRIOR TO SETTING THE REAL TIME CLOCK ALARM (INT 1AH, AH= 06H). :
442                                     <1> ;-----
443                                     <1>
444                                     <1> RTC_A_INT: ; 07/01/2017
445                                     <1> ;RTC_INT: ; ALARM INTERRUPT
446 000064A2 1E        <1>      push     ds ; LEAVE INTERRUPTS DISABLED
447 000064A3 50        <1>      push     eax ; SAVE REGISTERS
448 000064A4 57        <1>      push     edi
449                                     <1> RTC_I_1: ; CHECK FOR SECOND INTERRUPT
450 000064A5 66B88C8B <1>      mov     ax, 256*(CMOS_REG_B+NMI)+CMOS_REG_C+NMI ; ALARM AND STATUS
451 000064A9 E670      <1>      out     CMOS_PORT, al ; WRITE ALARM FLAG MASK ADDRESS
452 000064AB 90        <1>      nop ; I/O DELAY
453 000064AC EB00      <1>      jmp     short $+2
454 000064AE E471      <1>      in      al, CMOS_DATA ; READ AND RESET INTERRUPT REQUEST FLAGS
455 000064B0 A860      <1>      test    al, 01100000b ; CHECK FOR EITHER INTERRUPT PENDING
456 000064B2 745B      <1>      jz      short RTC_I_9 ; EXIT IF NOT A VALID RTC INTERRUPT
457                                     <1>
458 000064B4 86C4      <1>      xchg    ah, al ; SAVE FLAGS AND GET ENABLE ADDRESS
459 000064B6 E670      <1>      out     CMOS_PORT, al ; WRITE ALARM ENABLE MASK ADDRESS
460 000064B8 90        <1>      nop ; I/O DELAY
461 000064B9 EB00      <1>      jmp     short $+2
462 000064BB E471      <1>      in      al, CMOS_DATA ; READ CURRENT ALARM ENABLE MASK
463 000064BD 20E0      <1>      and     al, ah ; ALLOW ONLY SOURCES THAT ARE ENABLED
464 000064BF A840      <1>      test    al, 01000000b ; CHECK FOR PERIODIC INTERRUPT
465 000064C1 7439      <1>      jz      short RTC_I_5 ; SKIP IF NOT A PERIODIC INTERRUPT
466                                     <1>
467                                     <1> ;----- DECREMENT WAIT COUNT BY INTERRUPT INTERVAL
468                                     <1>
469 000064C3 66BF1000 <1>      mov     di, KDATA ; kernel data segment
470 000064C7 8EDF      <1>      mov     ds, di
471                                     <1>
472 000064C9 812D[18770100]D003- <1>      sub     dword [RTC_LH], 976 ; DECREMENT COUNT BY 1/1024
473 000064D1 0000      <1>
474 000064D3 7327      <1>      jnc     short RTC_I_5 ; SKIP TILL 32 BIT WORD LESS THAN ZERO
475                                     <1>
476                                     <1> ;----- TURN OFF PERIODIC INTERRUPT ENABLE
477                                     <1>
478                                     <1>      ;push     ax ; SAVE INTERRUPT FLAG MASK
479 000064D5 50        <1>      ; 18/04/2021
480 000064D6 66B88B8B <1>      push     eax
481 000064DA E670      <1>      mov     ax, 257*(CMOS_REG_B+NMI) ; INTERRUPT ENABLE REGISTER
482 000064DC 90        <1>      out     CMOS_PORT, al ; WRITE ADDRESS TO CMOS CLOCK
483 000064DD EB00      <1>      nop ; I/O DELAY
484 000064DF E471      <1>      jmp     short $+2
485 000064E1 248F      <1>      in      al, CMOS_DATA ; READ CURRENT ENABLES
486 000064E3 86E0      <1>      and     al, 0BFh ; TURN OFF PIE
487 000064E5 E670      <1>      xchg    al, ah ; GET CMOS ADDRESS AND SAVE VALUE
488 000064E7 86E0      <1>      out     CMOS_PORT, al ; ADDRESS REGISTER B
489 000064E9 E671      <1>      xchg    al, ah ; GET NEW INTERRUPT ENABLE MASK
490 000064EB C605[1C770100]00 <1>      out     CMOS_DATA, al ; SET MASK IN INTERRUPT ENABLE REGISTER
491 000064F2 8B3D[1D770100] <1>      mov     byte [RTC_WAIT_FLAG], 0 ; SET FUNCTION ACTIVE FLAG OFF
492 000064F4 C60780 <1>      mov     edi, [USER_FLAG] ; SET UP (DS:DI) TO POINT TO USER FLAG
493                                     <1>      ;mov     byte [edi], 80h ; TURN ON USERS FLAG
494                                     <1>      ;pop     ax ; GET INTERRUPT SOURCE BACK
495 000064FB 58        <1>      ; 18/04/2021
496                                     <1>      pop     eax
497 000064FC A820      <1>      test    al, 00100000b ; TEST FOR ALARM INTERRUPT
498 000064FE 740D      <1>      jz      short RTC_I_7 ; SKIP USER INTERRUPT CALL IF NOT ALARM
499                                     <1>
500 00006500 B00D      <1>      mov     al, CMOS_REG_D ; POINT TO DEFAULT READ ONLY REGISTER
501 00006502 E670      <1>      out     CMOS_PORT, al ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
502 00006504 FB        <1>      sti ; INTERRUPTS BACK ON NOW
503 00006505 52        <1>      push     edx

```

```

504 00006506 E8E3B60000 <1> call INT4Ah ; TRANSFER TO USER ROUTINE
505 0000650B 5A <1> pop edx
506 0000650C FA <1> cli ; BLOCK INTERRUPT FOR RETRY
507 <1> RTC_I_7: ; RESTART ROUTINE TO HANDLE DELAYED
508 0000650D EB96 <1> jmp short RTC_I_1 ; ENTRY AND SECOND EVENT BEFORE DONE
509 <1>
510 <1> RTC_I_9: ; EXIT - NO PENDING INTERRUPTS
511 0000650F B00D <1> mov al, CMOS_REG_D ; POINT TO DEFAULT READ ONLY REGISTER
512 00006511 E670 <1> out CMOS_PORT, al ; ENABLE NMI AND CMOS ADDRESS TO DEFAULT
513 00006513 B020 <1> mov al, EOI ; END OF INTERRUPT MASK TO 8259 - 2
514 00006515 E6A0 <1> out INTB00, al ; TO 8259 - 2
515 00006517 E620 <1> out INTA00, al ; TO 8259 - 1
516 00006519 5F <1> pop edi ; RESTORE REGISTERS
517 0000651A 58 <1> pop eax
518 0000651B 1F <1> pop ds
519 0000651C CF <1> iretd ; END OF INTERRUPT
520 <1>
521 <1> ;-----
522 <1>
523 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
524 <1> ; 22/08/2014 (Retro UNIX 386 v1)
525 <1> ; IBM PC/AT BIOS source code ----- 10/06/85 (bios2.asm)
526 <1> UPD_IPR: ; WAIT TILL UPDATE NOT IN PROGRESS
527 0000651D 51 <1> push ecx
528 <1>
529 <1> ; 29/05/2016
530 0000651E B968110000 <1> mov ecx, ((1984+244)*4)/2 ; AWARD BIOS 1999, ATIME.ASM
531 <1> ; 'WAITCPU_CHK_UD_STAT'
532 <1> ; (244us + 1984us)
533 <1> ; (assume each read takes
534 <1> ; 2 microseconds).
535 <1> ;mov ecx, 65535
536 <1> ;mov cx, 800 ; SET TIMEOUT LOOP COUNT (= 800)
537 <1> UPD_10:
538 00006523 B00A <1> mov al, CMOS_REG_A ; ADDRESS STATUS REGISTER A
539 00006525 FA <1> cli ; NO TIMER INTERRUPTS DURING UPDATES
540 00006526 E828000000 <1> call CMOS_READ ; READ UPDATE IN PROCESS FLAG
541 00006528 A880 <1> test al, 80h ; IF UIP BIT IS ON ( CANNOT READ TIME )
542 0000652D 7406 <1> jz short UPD_90 ; EXIT WITH CY= 0 IF CAN READ CLOCK NOW
543 0000652F FB <1> sti ; ALLOW INTERRUPTS WHILE WAITING
544 00006530 E2F1 <1> loop UPD_10 ; LOOP TILL READY OR TIMEOUT
545 00006532 31C0 <1> xor eax, eax ; xor ax, ax ; CLEAR RESULTS IF ERROR
546 00006534 F9 <1> stc ; SET CARRY FOR ERROR
547 <1> UPD_90:
548 00006535 59 <1> pop ecx ; RESTORE CALLERS REGISTER
549 00006536 FA <1> cli ; INTERRUPTS OFF DURING SET
550 00006537 C3 <1> retn ; RETURN WITH CY FLAG SET
551 <1>
552 <1> ; 18/04/2021
553 <1> RTC_STA: ; INITIALIZE REAL TIME CLOCK
554 <1> ;mov al, CMOS_REG_A ; ADDRESS REGISTER A AND LOAD DATA MASK
555 <1> ;mov ah, 26h
556 00006538 66B80A26 <1> mov ax, (26h*100h)+CMOS_REG_A
557 0000653C E8E7FEFFFF <1> call CMOS_WRITE ; INITIALIZE STATUS REGISTER A
558 <1> ;mov al, CMOS_REG_B ; SET "SET BIT" FOR CLOCK INITIALIZATION
559 <1> ;mov ah, 82h
560 00006541 66B80B82 <1> mov ax, (82h*100h)+CMOS_REG_B
561 00006545 E8DEF0FFFF <1> call CMOS_WRITE ; AND 24 HOUR MODE TO REGISTER B
562 0000654A B00C <1> mov al, CMOS_REG_C ; ADDRESS REGISTER C
563 0000654C E802000000 <1> call CMOS_READ ; READ REGISTER C TO INITIALIZE
564 00006551 B00D <1> mov al, CMOS_REG_D ; ADDRESS REGISTER D
565 <1> ; 18/04/2021
566 <1> ;call CMOS_READ ; READ REGISTER D TO INITIALIZE
567 <1> ;retn
568 <1> ;jmp short CMOS_READ ; 18/04/2021
569 <1>
570 <1> ;-----
571 <1>
572 <1> ; 29/07/2022 - TRDOS 386 v2.0.5
573 <1> ; 18/04/2021 - TRDOS 386 v2.0.4
574 <1> ; 29/05/2016 - TRDOS 386 (TRDOS v2.0)
575 <1> ; 22/08/2014 (Retro UNIX 386 v1)
576 <1> ; IBM PC/AT BIOS source code ----- 10/06/85 (test4.asm)
577 <1>
578 <1> ---- CMOS_READ -----
579 <1> ; READ BYTE FROM CMOS_SYSTEM CLOCK CONFIGURATION TABLE :
580 <1> ; :
581 <1> ; INPUT: (AL)= CMOS_TABLE ADDRESS TO BE READ :
582 <1> ; BIT 7 = 0 FOR NMI ENABLED AND 1 FOR NMI DISABLED ON EXIT :
583 <1> ; BITS 6-0 = ADDRESS OF TABLE LOCATION TO READ :
584 <1> ; :
585 <1> ; OUTPUT: (AL) VALUE AT LOCATION (AL) MOVED INTO (AL). IF BIT 7 OF (AL) WAS :
586 <1> ; ON THEN NMI LEFT DISABLED, DURING THE CMOS READ BOTH NMI AND :
587 <1> ; NORMAL INTERRUPTS ARE DISABLED TO PROTECT CMOS DATA INTEGRITY. :
588 <1> ; THE CMOS ADDRESS REGISTER IS POINTED TO A DEFAULT VALUE AND :
589 <1> ; THE INTERRUPT FLAG RESTORED TO THE ENTRY STATE ON RETURN. :
590 <1> ; ONLY THE (AL) REGISTER AND THE NMI STATE IS CHANGED. :
591 <1> ;-----
592 <1>
593 <1> CMOS_READ:
594 <1> ; 29/07/2022 (TRDOS 386 Kernel v2.0.5)
595 <1> ;pushf ; SAVE INTERRUPT ENABLE STATUS AND FLAGS
596 00006553 D0C0 <1> rol al, 1 ; MOVE NMI BIT TO LOW POSITION
597 00006555 F9 <1> stc ; FORCE NMI BIT ON IN CARRY FLAG
598 00006556 D0D8 <1> rcr al, 1 ; HIGH BIT ON TO DISABLE NMI - OLD IN CY
599 00006558 FA <1> cli ; DISABLE INTERRUPTS
600 00006559 E670 <1> out CMOS_PORT, al ; ADDRESS LOCATION AND DISABLE NMI
601 <1> ; 29/05/2016
602 <1> ;nop ; I/O DELAY
603 0000655B E6EB <1> out 0EBh, al ; NEWIODELAY ; AWARD BIOS 1999, ATIME.ASM
604 <1> ;
605 0000655D E471 <1> in al, CMOS_DATA ; READ THE REQUESTED CMOS LOCATION
606 <1> ;push ax ; SAVE (AH) REGISTER VALUE AND CMOS BYTE
607 <1> ; 18/04/2021
608 0000655F 50 <1> push eax
609 <1> ; 15/03/2015 ; IBM PC/XT Model 286 BIOS source code
610 <1> ; ----- 10/06/85 (test4.asm)
611 00006560 B01E <1> mov al, CMOS_SHUT_DOWN*2 ; GET ADDRESS OF DEFAULT LOCATION
612 <1> ;mov al, CMOS_REG_D*2 ; GET ADDRESS OF DEFAULT LOCATION
613 00006562 D0D8 <1> rcr al, 1 ; PUT ORIGINAL NMI MASK BIT INTO ADDRESS
614 00006564 E670 <1> out CMOS_PORT, al ; SET DEFAULT TO READ ONLY REGISTER
615 <1> ;pop ax ; RESTORE (AH) AND (AL), CMOS BYTE
616 <1> ; 18/04/2021
617 00006566 58 <1> pop eax
618 <1> ; 29/07/2022
619 <1> ;popf
620 00006567 F8 <1> clc ; 29/07/2022
621 00006568 C3 <1> retn ; RETURN WITH FLAGS RESTORED
622 <1>
623 <1> ;-----
624 <1>
625 <1> ; /// End of TIMER FUNCTIONS ///
3230
3231 00006569 90<rep 7h> Align 16

```



```

3232
3233
3234
3235
3236
3237
3238
3239
3240
3241 00006570 0000000000000000
3242
3243
3244
3245
3246 00006578 FFFF0000009ACF00
3247
3248
3249 00006580 FFFF00000092CF00
3250
3251
3252
3253 00006588 FFFB000040FACF00
3254
3255
3256 00006590 FFFB000040F2CF00
3257
3258
3259 00006598 6700
3260
3261
3262 0000659A 0000
3263
3264 0000659C 00
3265
3266 0000659D E9
3267 0000659E 00
3268
3269 0000659F 00
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280 000065A0 FFFF0000069A0000
3281
3282
3283
3284 000065A8 FF05000000920000
3285
3286
3287
3288 000065B0 FFFF00000A920000
3289
3290
3291
3292 000065B8 FFFF00000B920000
3293
3294
3295
3296 000065C0 FF7F00800B920000
3297
3298
3299
3300
3301
3302 000065C8 FFFF000006920000
3303
3304
3305
3306 000065D0 FF03000000920000
3307
3308
3309
3310 000065D8 FF07000000920000
3311
3312
3313
3314 000065E0 FFFF0000009AFF00
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355

gdt:; Global Descriptor Table
; 02/12/2020
; (30/07/2015, conforming cs)
; (26/03/2015)
; (24/03/2015, tss)
; (19/03/2015)
; (29/12/2013)
;
; dw 0, 0, 0, 0 ; NULL descriptor
gdt_kcode:
; 18/08/2014
; 8h kernel code segment, base = 00000000h
; dw 0FFFFh, 0, 9E00h, 00CFh ; KCODE ; 30/12/2016
; dw 0FFFFh, 0, 9A00h, 00CFh ; KCODE
gdt_kdata:
; 10h kernel data segment, base = 00000000h
; dw 0FFFFh, 0, 9200h, 00CFh ; KDATA
gdt_ucode:
; 18h user code segment, base address = 400000h ; CORE
; dw 0FBFFh, 0, 0FE40h, 00CFh ; UCODE ; 30/12/2016
; dw 0FBFFh, 0, 0FA40h, 00CFh ; UCODE
gdt_udata:
; 23h user data segment, base address = 400000h ; CORE
; dw 0FBFFh, 0, 0F240h, 00CFh ; UDATA
gdt_tss:
; Task State Segment
; dw 0067h ; Limit = 103 ; (104-1, tss size = 104 byte,
; ; no IO permission in ring 3)
gdt_tss0:
; dw 0 ; TSS base address, bits 0-15
gdt_tss1:
; db 0 ; TSS base address, bits 16-23
; 49h
; db 11101001b ; 0E9h => P=1/DPL=11/0/1/0/B/1 --> B = Task is busy (1)
; db 0 ; G/0/0/AVL/LIMIT=0000 ; (Limit bits 16-19 = 0000) (G=0, 1 byte)
gdt_tss2:
; db 0 ; TSS base address, bits 24-31
;
; 04/12/2023 - TRDOS v2.0.7
; 30/11/2020
; 29/11/2020 - TRDOS v2.0.3
; VESA VBE3 VIDE BIOS 32 BIT PMI SEGMENTS (16 bit segments)
; 30h ; VBE3CS
_vbe3_CS: ; vesa vbe3 bios uses this as code seg (same addr with _vbe3_DS)
; limit = 65536, base addr = 0, P/DPL/1/Type/C/R/A = 9Ah, 16 bit
; dw 0FFFFh, 0, 9A00h, 0 ; Note: base addr will be initialized
; 04/12/2023 - TRDOS 386 v2.0.7 ; (!*!+)
; dw 0FFFFh, 0, 9A06h, 0 ; VBE3BIOSCODE_ADDR = 60000h
; 38h ; VBE3BDS
_vbe3_BDS: ; vesa vbe3 bios uses this as equivalent of rombios data segment
; limit = 1536, base addr = 0, P/DPL/1/Type/E/W/A = 92h, 16 bit
; dw 05FFh, 0, 9200h, 0 ; Note: base addr will be initialized
; 40h ; VBE3A000
_A0000Sel: ; VGA default video memory address
; limit = 65536, base addr = 0A0000h, 16 bit
; dw 0FFFFh, 0, 920Ah, 0
; 48h ; VBE3B000
_B0000Sel: ; MDA (monochrome) video memory address
; limit = 65536, base addr = 0B0000h, 16 bit
; dw 0FFFFh, 0, 920Bh, 0
; 50h ; VBE3B800
_B8000Sel: ; CGA video memory address
; limit = 32768, base addr = 0B8000h, 16 bit
; dw 07FFFh, 8000h, 920Bh, 0
; 58h ; VBE3DS
_vbe3_DS: ; vesa vbe3 bios uses this as data seg (CodeSegSel in PMInfoBlock)
; limit = 65536, base addr = 0, P/DPL/1/Type/E/W/A = 92h, 16 bit
; dw 0FFFFh, 0, 9200h, 0 ; Note: base addr will be initialized
; 04/12/2023 - TRDOS 386 v2.0.7 ; (!*!+)
; dw 0FFFFh, 0, 9206h, 0 ; VBE3BIOSCODE_ADDR = 60000h
; 60h ; VBE3SS
_vbe3_SS: ; kernel's stack segment but 16 bit version (same stack addr)
; limit = 1024, base addr = 0, P/DPL/1/Type/E/W/A = 92h, 16 bit
; dw 03FFh, 0, 9200h, 0 ; Note: base addr will be initialized
; 68h ; VBE3ES
_vbe3_ES: ; extra 16 bit segment points to buffers in kernel's mem space
; limit = 2048, base addr = 0, P/DPL/1/Type/E/W/A = 92h, 16 bit
; dw 07FFh, 0, 9200h, 0 ; Note: base addr will be initialized
; 70h ; KODE16
_16bit_CS: ; 16 bit code segment points to kernel's far return addr
; limit = 16M, base addr = 0, P/DPL/1/Type/E/W/A = 92h, 16 bit
; dw 0FFFFh, 0, 9A00h, 00FFh ; Note: base addr will be initialized
gdt_end:
; 9Eh = 1001 1110b (GDT byte 5) P=1/DPL=00/1/TYPE=1110,
; ; Type= 1 (code)/C=1/R=1/A=0
; P= Present, DPL=0=ring 0, 1= user (0= system)
; 1= Code C= Conforming, R= Readable, A= Accessed
;
; 9Ah = 1001 1010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
; ; Type= 1 (code)/C=0/R=1/A=0
; P= Present, DPL=0=ring 0, 1= user (0= system)
; 1= Code C= non-Conforming, R= Readable, A= Accessed
;
; 92h = 1001 0010b (GDT byte 5) P=1/DPL=00/1/TYPE=1010,
; ; Type= 0 (data)/E=0/W=1/A=0
; P= Present, DPL=0=ring 0, 1= user (0= system)
; 0= Data E= Expansion direction (1= down, 0= up)
; W= Writeable, A= Accessed
;
; FEh = 1111 1110b (GDT byte 5) P=1/DPL=11/1/TYPE=1110,
; ; Type= 1 (code)/C=1/R=1/A=0
; P= Present, DPL=3=ring 3, 1= user (0= system)
; 1= Code C= Conforming, R= Readable, A= Accessed
;
; FAh = 1111 1010b (GDT byte 5) P=1/DPL=11/1/TYPE=1010,
; ; Type= 1 (code)/C=0/R=1/A=0
; P= Present, DPL=3=ring 3, 1= user (0= system)
; 1= Code C= non-Conforming, R= Readable, A= Accessed
;
; F2h = 1111 0010b (GDT byte 5) P=1/DPL=11/1/TYPE=0010,
; ; Type= 0 (data)/E=0/W=1/A=0
; P= Present, DPL=3=ring 3, 1= user (0= system)
; 0= Data E= Expansion direction (1= down, 0= up)
;
; CFh = 1100 1111b (GDT byte 6) G=1/B=1/0/AVL=0, Limit=1111b (3)
;
; Limit = FFFFh (=> FFFFh+1= 100000h) // bits 0-15, 48-51 //
; = 100000h * 1000h (G=1) = 4GB
; Limit = FFBFFh (=> FFBFFh+1= FFC00h) // bits 0-15, 48-51 //
; = FFC00h * 1000h (G=1) = 4GB - 4MB
; G= Granularity (1= 4KB), B= Big (32 bit),
; AVL= Available to programmers

```

```

3356
3357
3358 000065E8 7700
3359 000065EA [70650000]
3360
3361
3362
3363 000065EE 7F02
3364 000065F0 [B8730100]
3365
3366
3367
3368
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 000065F4 [57660000]
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111

gdttd:
    dw gdt_end - gdt - 1    ; Limit (size)
    dd gdt                  ; Address of the GDT

    ; 20/08/2014
idtd:
    dw idt_end - idt - 1    ; Limit (size)
    dd idt                  ; Address of the IDT

    ; 20/02/2017
    ; 11/03/2015
%include 'diskdata.s'      ; DISK (BIOS) DATA (initialized)
; *****
; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.5 - diskdata.s
; -----
; Last Update: 06/08/2022 (Previous: 24/01/2016)
; -----
; Beginning: 24/01/2016
; -----
; Assembler: NASM version 2.15 (trdos386.s)
; -----
; Turkish Rational DOS
; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
; -----
; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
; diskdata.inc (11/03/2015)
; -----
; Derived from 'IBM PC-XT-286' BIOS source code (1986)
; *****
; Retro UNIX 386 v1 Kernel - DISKDATA.INC
; Last Modification: 11/03/2015
; (Initialized Disk Parameters Data section for 'DISKIO.INC')
; -----
; 80286 INTERRUPT LOCATIONS      :
; REFERENCED BY POST & BIOS      :
; -----
DISK_POINTER:    dd    MD_TBL6    ; Pointer to Diskette Parameter Table
; -----
; IBM PC-XT Model 286 source code ORGS.ASM (06/10/85) - 14/12/2014
; -----
; DISK_BASE                      :
; THIS IS THE SET OF PARAMETERS REQUIRED FOR      :
; DISKETTE OPERATION. THEY ARE POINTED AT BY THE :
; DATA VARIABLE @DISK_POINTER. TO MODIFY THE PARAMETERS, :
; BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT :
; -----
; DISK_BASE:
; DB    11011111B    ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
; DB    2            ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
; DB    MOTOR_WAIT    ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
; DB    2            ; 512 BYTES/SECTOR
; ;DB    15          ; EOT (LAST SECTOR ON TRACK)
; db    18            ; (EOT for 1.44MB diskette)
; DB    01BH          ; GAP LENGTH
; DB    0FFH          ; DTL
; ;DB    054H          ; GAP LENGTH FOR FORMAT
; db    06ch          ; (for 1.44MB diskette)
; DB    0F6H          ; FILL BYTE FOR FORMAT
; DB    15            ; HEAD SETTLE TIME (MILLISECONDS)
; DB    8             ; MOTOR START TIME (1/8 SECONDS)
; -----
; ROM BIOS DATA AREAS          :
; -----
; DATA          SEGMENT AT 40H    ; ADDRESS= 0040:0000
; @EQUIP_FLAG    DW    ?          ; INSTALLED HARDWARE FLAGS
; -----
; DISKETTE DATA AREAS          :
; -----
; @SEEK_STATUS    DB    ?          ; DRIVE RECALIBRATION STATUS
; ; BIT 3-0 = DRIVE 3-0 RECALIBRATION
; ; BEFORE NEXT SEEK IF BIT IS = 0
; @MOTOR_STATUS    DB    ?          ; MOTOR STATUS
; ; BIT 3-0 = DRIVE 3-0 CURRENTLY RUNNING
; ; BIT 7 = CURRENT OPERATION IS A WRITE
; @MOTOR_COUNT    DB    ?          ; TIME OUT COUNTER FOR MOTOR(S) TURN OFF
; @DSKETTE_STATUS DB    ?          ; RETURN CODE STATUS BYTE
; ; CMD_BLOCK IN STACK FOR DISK OPERATION
; @NEC_STATUS     DB    7 DUP(?)    ; STATUS BYTES FROM DISKETTE OPERATION
; -----
; POST AND BIOS WORK DATA AREA :
; -----
; @INTR_FLAG DB    ?          ; FLAG INDICATING AN INTERRUPT HAPPENED
; -----
; TIMER DATA AREA              :
; -----
; 17/12/2014 (IRQ 0 - INT 08H)
; TIMER_LOW equ    46Ch          ; Timer ticks (counter) @ 40h:006Ch
; TIMER_HIGH equ   46Eh          ; (18.2 timer ticks per second)
; TIMER_OFL equ    470h          ; Timer - 24 hours flag @ 40h:0070h
; -----
; ADDITIONAL MEDIA DATA        :
; -----
; @LAstrate DB    ?          ; LAST DISKETTE DATA RATE SELECTED
; @DSK_STATE DB    ?          ; DRIVE 0 MEDIA STATE
; ; DB    ?          ; DRIVE 1 MEDIA STATE
; ; DB    ?          ; DRIVE 0 OPERATION START STATE
; ; DB    ?          ; DRIVE 1 OPERATION START STATE
; @DSK_TRK    DB    ?          ; DRIVE 0 PRESENT CYLINDER
; ; DB    ?          ; DRIVE 1 PRESENT CYLINDER
; -----
; DATA          ENDS          ; END OF BIOS DATA SEGMENT
; -----
; DRIVE TYPE TABLE              :
; -----
; 16/02/2015 (unix386.s, 32 bit modifications)
DR_TYPE:

```

```

112 000065F8 01      <1>          DB      01          ; DRIVE TYPE, MEDIA TABLE
113                  <1>          ;DW      MD_TBL1
114 000065F9 [16660000] <1>          dd      MD_TBL1
115 000065FD 82      <1>          DB      02+BIT7ON
116                  <1>          ;DW      MD_TBL2
117 000065FE [23660000] <1>          dd      MD_TBL2
118 00006602 02      <1> DR_DEFAULT: DB      02
119                  <1>          ;DW      MD_TBL3
120 00006603 [30660000] <1>          dd      MD_TBL3
121 00006607 03      <1>          DB      03
122                  <1>          ;DW      MD_TBL4
123 00006608 [3D660000] <1>          dd      MD_TBL4
124 0000660C 84      <1>          DB      04+BIT7ON
125                  <1>          ;DW      MD_TBL5
126 0000660D [4A660000] <1>          dd      MD_TBL5
127 00006611 04      <1>          DB      04
128                  <1>          ;DW      MD_TBL6
129 00006612 [57660000] <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> -----
135                  <1> ; MEDIA/DRIVE PARAMETER TABLES :
136                  <1> -----
137                  <1> ; 360 KB MEDIA IN 360 KB DRIVE :
138                  <1> -----
139                  <1> MD_TBL1:
140                  <1> DB      11011111B      ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
141                  <1> DB      2          ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
142                  <1> DB      MOTOR_WAIT    ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
143                  <1> DB      2          ; 512 BYTES/SECTOR
144                  <1> DB      09          ; EOT (LAST SECTOR ON TRACK)
145                  <1> DB      02AH        ; GAP LENGTH
146                  <1> DB      0FFH        ; DTL
147                  <1> DB      050H        ; GAP LENGTH FOR FORMAT
148                  <1> DB      0F6H        ; FILL BYTE FOR FORMAT
149                  <1> DB      15          ; HEAD SETTLE TIME (MILLISECONDS)
150                  <1> DB      8          ; MOTOR START TIME (1/8 SECONDS)
151                  <1> DB      39          ; MAX. TRACK NUMBER
152                  <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                  <1> DB      11011111B      ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
158                  <1> DB      2          ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
159                  <1> DB      MOTOR_WAIT    ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
160                  <1> DB      2          ; 512 BYTES/SECTOR
161                  <1> DB      09          ; EOT (LAST SECTOR ON TRACK)
162                  <1> DB      02AH        ; GAP LENGTH
163                  <1> DB      0FFH        ; DTL
164                  <1> DB      050H        ; GAP LENGTH FOR FORMAT
165                  <1> DB      0F6H        ; FILL BYTE FOR FORMAT
166                  <1> DB      15          ; HEAD SETTLE TIME (MILLISECONDS)
167                  <1> DB      8          ; MOTOR START TIME (1/8 SECONDS)
168                  <1> DB      39          ; MAX. TRACK NUMBER
169                  <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                  <1> DB      11011111B      ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
175                  <1> DB      2          ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
176                  <1> DB      MOTOR_WAIT    ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
177                  <1> DB      2          ; 512 BYTES/SECTOR
178                  <1> DB      15          ; EOT (LAST SECTOR ON TRACK)
179                  <1> DB      01BH        ; GAP LENGTH
180                  <1> DB      0FFH        ; DTL
181                  <1> DB      054H        ; GAP LENGTH FOR FORMAT
182                  <1> DB      0F6H        ; FILL BYTE FOR FORMAT
183                  <1> DB      15          ; HEAD SETTLE TIME (MILLISECONDS)
184                  <1> DB      8          ; MOTOR START TIME (1/8 SECONDS)
185                  <1> DB      79          ; MAX. TRACK NUMBER
186                  <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                  <1> DB      11011111B      ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
192                  <1> DB      2          ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
193                  <1> DB      MOTOR_WAIT    ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
194                  <1> DB      2          ; 512 BYTES/SECTOR
195                  <1> DB      09          ; EOT (LAST SECTOR ON TRACK)
196                  <1> DB      02AH        ; GAP LENGTH
197                  <1> DB      0FFH        ; DTL
198                  <1> DB      050H        ; GAP LENGTH FOR FORMAT
199                  <1> DB      0F6H        ; FILL BYTE FOR FORMAT
200                  <1> DB      15          ; HEAD SETTLE TIME (MILLISECONDS)
201                  <1> DB      8          ; MOTOR START TIME (1/8 SECONDS)
202                  <1> DB      79          ; MAX. TRACK NUMBER
203                  <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                  <1> DB      11011111B      ; SRT=D, HD UNLOAD=0F - 1ST SPECIFY BYTE
209                  <1> DB      2          ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
210                  <1> DB      MOTOR_WAIT    ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
211                  <1> DB      2          ; 512 BYTES/SECTOR
212                  <1> DB      09          ; EOT (LAST SECTOR ON TRACK)
213                  <1> DB      02AH        ; GAP LENGTH
214                  <1> DB      0FFH        ; DTL
215                  <1> DB      050H        ; GAP LENGTH FOR FORMAT
216                  <1> DB      0F6H        ; FILL BYTE FOR FORMAT
217                  <1> DB      15          ; HEAD SETTLE TIME (MILLISECONDS)
218                  <1> DB      8          ; MOTOR START TIME (1/8 SECONDS)
219                  <1> DB      79          ; MAX. TRACK NUMBER
220                  <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                  <1> DB      10101111B      ; SRT=A, HD UNLOAD=0F - 1ST SPECIFY BYTE
226                  <1> DB      2          ; HD LOAD=1, MODE=DMA - 2ND SPECIFY BYTE
227                  <1> DB      MOTOR_WAIT    ; WAIT TIME AFTER OPERATION TILL MOTOR OFF
228                  <1> DB      2          ; 512 BYTES/SECTOR
229                  <1> DB      18          ; EOT (LAST SECTOR ON TRACK)
230                  <1> DB      01BH        ; GAP LENGTH
231                  <1> DB      0FFH        ; DTL
232                  <1> DB      06CH        ; GAP LENGTH FOR FORMAT
233                  <1> DB      0F6H        ; FILL BYTE FOR FORMAT
234                  <1> DB      15          ; HEAD SETTLE TIME (MILLISECONDS)
235                  <1> DB      8          ; MOTOR START TIME (1/8 SECONDS)

```

```

236 00006662 4F      <1>      DB      79          ; MAX. TRACK NUMBER
237 00006663 00      <1>      DB      RATE_500      ; DATA TRANSFER RATE
238
239
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      <1>      db      NO_ERR
280      <1>      db      BAD_ADDR_MARK,BAD_SEEK,BAD_CMD,UNDEF_ERR
281      <1>      db      RECORD_NOT_FND,UNDEF_ERR,BAD_ECC,BAD_SECTOR
282      <1>      ;
283      <1>      ; 06/08/2022
284      <1>      ; 17/12/2014 (mov ax, [cfd])
285      <1>      ; 11/12/2014
286      <1>      ; cfd:      db      0          ; current floppy drive (for GET_PARM)
287      <1>      ; 17/12/2014          ; instead of 'DISK_POINTER'
288      <1>      ; pfd:      db      1          ; previous floppy drive (for GET_PARM)
289      <1>      ; (initial value of 'pfd
290      <1>      ; must be different then 'cfd' value
291      <1>      ; to force updating/initializing
292      <1>      ; current drive parameters)
293      <1>      ; 06/08/2022 - TRDOS 386 Kernel v2.0.5
294      <1>      ; 11/07/2022 - Retro UNIX 386 v1.1 (kernel v0.2.1.5)
295      <1>      pfd:      db      0FFh
296      <1>      ;
297      <1>      align 2
298      <1>      ;
299      <1>      HF_PORT:      dw      1F0h      ; Default = 1F0h
300      <1>      ; (170h)
301      <1>      HF_REG_PORT:      dw      3F6h      ; HF_PORT + 206h
302      <1>      ;
303      <1>      ; 05/01/2015
304      <1>      hf_m_s:      db      0          ; (0 = Master, 1 = Slave)
305      <1>      ;
306      <1>      ; *****
3369
3370      <1>      Align 2
3371      <1>      ;
3372      <1>      ; 04/11/2014 (Retro UNIX 386 v1)
3373      <1>      mem_lm_1k:      dw 0      ; Number of contiguous KB between
3374      <1>      ; 1 and 16 MB, max. 3C00h = 15 MB.
3375      <1>      mem_16m_64k:      dw 0      ; Number of contiguous 64 KB blocks
3376      <1>      ; between 16 MB and 4 GB.
3377      <1>      ;
3378      <1>      ; 12/11/2014 (Retro UNIX 386 v1)
3379      <1>      boot_drv:      db 0      ; boot drive number (physical)
3380      <1>      ; 24/11/2014
3381      <1>      drv:      db 0
3382      <1>      last_drv:      db 0      ; last hdd
3383      <1>      hdc:      db 0      ; number of hard disk drives
3384      <1>      ; (present/detected)
3385      <1>      ;
3386      <1>      ; 24/11/2014 (Retro UNIX 386 v1)
3387      <1>      ; Physical drive type & flags
3388      <1>      fd0_type:      db 0      ; floppy drive type
3389      <1>      fd1_type:      db 0      ; 4 = 1.44 Mb, 80 track, 3.5" (18 spt)
3390      <1>      ; 6 = 2.88 Mb, 80 track, 3.5" (36 spt)
3391      <1>      ; 3 = 720 kb, 80 track, 3.5" (9 spt)
3392      <1>      ; 2 = 1.2 Mb, 80 track, 5.25" (15 spt)
3393      <1>      ; 1 = 360 Kb, 40 track, 5.25" (9 spt)
3394      <1>      hd0_type:      db 0      ; EDD status for hd0 (bit 7 = present flag)
3395      <1>      hd1_type:      db 0      ; EDD status for hd1 (bit 7 = present flag)
3396      <1>      hd2_type:      db 0      ; EDD status for hd2 (bit 7 = present flag)
3397      <1>      hd3_type:      db 0      ; EDD status for hd3 (bit 7 = present flag)
3398      <1>      ; bit 0 - Fixed disk access subset supported
3399      <1>      ; bit 1 - Drive locking and ejecting
3400      <1>      ; bit 2 - Enhanced disk drive support
3401      <1>      ; bit 3 = Reserved (64 bit EDD support)
3402      <1>      ; (If bit 0 is '1' Retro UNIX 386 v1
3403      <1>      ; will interpret it as 'LBA ready'!)
3404      <1>      ;
3405      <1>      ; 08/08/2022
3406      <1>      ; (drv.cylinders, drv.spt, drv.spt will not be used now on)
3407      <1>      ; ('diskio.inc')
3408      <1>      ; ((spt and heads and cylinder counts will be taken from DPT))
3409      <1>      ;
3410      <1>      ; 11/03/2015 - 10/07/2015
3411      <1>      ; drv.cylinders:      dw 0,0,0,0,0,0,0
3412      <1>      ; drv.heads:      dw 0,0,0,0,0,0,0
3413      <1>      ; drv.spt:      dw 0,0,0,0,0,0,0
3414      <1>      ; 12/07/2022 - 11/03/2015
3415      <1>      drv.size:      dd 0,0,0,0,0,0,0
3416      <1>      ;
3417      <1>      ;
3418      <1>      ;
3419      <1>      ;
3420      <1>      ;
3421      <1>      ;
3422      <1>      ;
3423      <1>      ;
3424      <1>      ;
3425      <1>      ;
3426      <1>      ;
3427      <1>      ;
3428      <1>      ;
3429      <1>      ;
3430      <1>      ;
3431      <1>      ;
3432      <1>      ;
3433      <1>      ;
3434      <1>      ;
3435      <1>      ;
3436      <1>      ;
3437      <1>      ;
3438      <1>      ;
3439      <1>      ;
3440      <1>      ;
3441      <1>      ;
3442      <1>      ;
3443      <1>      ;
3444      <1>      ;
3445      <1>      ;
3446      <1>      ;
3447      <1>      ;
3448      <1>      ;
3449      <1>      ;
3450      <1>      ;
3451      <1>      ;
3452      <1>      ;
3453      <1>      ;
3454      <1>      ;
3455      <1>      ;
3456      <1>      ;
3457      <1>      ;
3458      <1>      ;
3459      <1>      ;
3460      <1>      ;
3461      <1>      ;
3462      <1>      ;
3463      <1>      ;
3464      <1>      ;
3465      <1>      ;
3466      <1>      ;
3467      <1>      ;
3468      <1>      ;
3469      <1>      ;
3470      <1>      ;
3471      <1>      ;
3472      <1>      ;
3473      <1>      ;
3474      <1>      ;
3475      <1>      ;
3476      <1>      ;
3477      <1>      ;
3478      <1>      ;
3479      <1>      ;
3480      <1>      ;
3481      <1>      ;
3482      <1>      ;
3483      <1>      ;
3484      <1>      ;
3485      <1>      ;
3486      <1>      ;
3487      <1>      ;
3488      <1>      ;
3489      <1>      ;
3490      <1>      ;
3491      <1>      ;
3492      <1>      ;
3493      <1>      ;
3494      <1>      ;
3495      <1>      ;
3496      <1>      ;
3497      <1>      ;
3498      <1>      ;
3499      <1>      ;
3500      <1>      ;
3501      <1>      ;
3502      <1>      ;
3503      <1>      ;
3504      <1>      ;
3505      <1>      ;
3506      <1>      ;
3507      <1>      ;
3508      <1>      ;
3509      <1>      ;
3510      <1>      ;
3511      <1>      ;
3512      <1>      ;
3513      <1>      ;
3514      <1>      ;
3515      <1>      ;
3516      <1>      ;
3517      <1>      ;
3518      <1>      ;
3519      <1>      ;
3520      <1>      ;
3521      <1>      ;
3522      <1>      ;
3523      <1>      ;
3524      <1>      ;
3525      <1>      ;
3526      <1>      ;
3527      <1>      ;
3528      <1>      ;
3529      <1>      ;
3530      <1>      ;
3531      <1>      ;
3532      <1>      ;
3533      <1>      ;
3534      <1>      ;
3535      <1>      ;
3536      <1>      ;
3537      <1>      ;
3538      <1>      ;
3539      <1>      ;
3540      <1>      ;
3541      <1>      ;
3542      <1>      ;
3543      <1>      ;
3544      <1>      ;
3545      <1>      ;
3546      <1>      ;
3547      <1>      ;
3548      <1>      ;
3549      <1>      ;
3550      <1>      ;
3551      <1>      ;
3552      <1>      ;
3553      <1>      ;
3554      <1>      ;
3555      <1>      ;
3556      <1>      ;
3557      <1>      ;
3558      <1>      ;
3559      <1>      ;
3560      <1>      ;
3561      <1>      ;
3562      <1>      ;
3563      <1>      ;
3564      <1>      ;
3565      <1>      ;
3566      <1>      ;
3567      <1>      ;
3568      <1>      ;
3569      <1>      ;
3570      <1>      ;
3571      <1>      ;
3572      <1>      ;
3573      <1>      ;
3574      <1>      ;
3575      <1>      ;
3576      <1>      ;
3577      <1>      ;
3578      <1>      ;
3579      <1>      ;
3580      <1>      ;
3581      <1>      ;
3582      <1>      ;
3583      <1>      ;
3584      <1>      ;
3585      <1>      ;
3586      <1>      ;
3587      <1>      ;
3588      <1>      ;
3589      <1>      ;
3590      <1>      ;
3591      <1>      ;
3592      <1>      ;
3593      <1>      ;
3594      <1>      ;
3595      <1>      ;
3596      <1>      ;
3597      <1>      ;
3598      <1>      ;
3599      <1>      ;
3600      <1>      ;
3601      <1>      ;
3602      <1>      ;
3603      <1>      ;
3604      <1>      ;
3605      <1>      ;
3606      <1>      ;
3607      <1>      ;
3608      <1>      ;
3609      <1>      ;
3610      <1>      ;
3611      <1>      ;
3612      <1>      ;
3613      <1>      ;
3614      <1>      ;
3615      <1>      ;
3616      <1>      ;
3617      <1>      ;
3618      <1>      ;
3619      <1>      ;
3620      <1>      ;
3621      <1>      ;
3622      <1>      ;
3623      <1>      ;
3624      <1>      ;
3625      <1>      ;
3626      <1>      ;
3627      <1>      ;
3628      <1>      ;
3629      <1>      ;
3630      <1>      ;
3631      <1>      ;
3632      <1>      ;
3633      <1>      ;
3634      <1>      ;
3635      <1>      ;
3636      <1>      ;
3637      <1>      ;
3638      <1>      ;
3639      <1>      ;
3640      <1>      ;
3641      <1>      ;
3642      <1>      ;
3643      <1>      ;
3644      <1>      ;
3645      <1>      ;
3646      <1>      ;
3647      <1>      ;
3648      <1>      ;
3649      <1>      ;
3650      <1>      ;
3651      <1>      ;
3652      <1>      ;
3653      <1>      ;
3654      <1>      ;
3655      <1>      ;
3656      <1>      ;
3657      <1>      ;
3658      <1>      ;
3659      <1>      ;
3660      <1>      ;
3661      <1>      ;
3662      <1>      ;
3663      <1>      ;
3664      <1>      ;
3665      <1>      ;
3666      <1>      ;
3667      <1>      ;
3668      <1>      ;
3669      <1>      ;
3670      <1>      ;
3671      <1>      ;
3672      <1>      ;
3673      <1>      ;
3674      <1>      ;
3675      <1>      ;
3676      <1>      ;
3677      <1>      ;
3678      <1>      ;
3679      <1>      ;
3680      <1>      ;
3681      <1>      ;
3682      <1>      ;
3683      <1>      ;
3684      <1>      ;
3685      <1>      ;
3686      <1>      ;
3687      <1>      ;
3688      <1>      ;
3689      <1>      ;
3690      <1>      ;
3691      <1>      ;
3692      <1>      ;
3693      <1>      ;
3694      <1>      ;
3695      <1>      ;
3696      <1>      ;
3697      <1>      ;
3698      <1>      ;
3699      <1>      ;
3700      <1>      ;
3701      <1>      ;
3702      <1>      ;
3703      <1>      ;
3704      <1>      ;
3705      <1>      ;
3706      <1>      ;
3707      <1>      ;
3708      <1>      ;
3709      <1>      ;
3710      <1>      ;
3711      <1>      ;
3712      <1>      ;
3713      <1>      ;
3714      <1>      ;
3715      <1>      ;
3716      <1>      ;
3717      <1>      ;
3718      <1>      ;
3719      <1>      ;
3720      <1>      ;
3721      <1>      ;
3722      <1>      ;
3723      <1>      ;
3724      <1>      ;
3725      <1>      ;
3726      <1>      ;
3727      <1>      ;
3728      <1>      ;
3729      <1>      ;
3730      <1>      ;
3731      <1>      ;
3732      <1>      ;
3733      <1>      ;
3734      <1>      ;
3735      <1>      ;
3736      <1>      ;
3737      <1>      ;
3738      <1>      ;
3739      <1>      ;
3740      <1>      ;
3741      <1>      ;
3742      <1>      ;
3743      <1>      ;
3744      <1>      ;
3745      <1>      ;
3746      <1>      ;
3747      <1>      ;
3748      <1>      ;
3749      <1>      ;
3750      <1>      ;
3751      <1>      ;
3752      <1>      ;
3753      <1>      ;
3754      <1>      ;
3755      <1>      ;
3756      <1>      ;
3757      <1>      ;
3758      <1>      ;
3759      <1>      ;
3760      <1>      ;
3761      <1>      ;
3762      <1>      ;
3763      <1>      ;
3764      <1>      ;
3765      <1>      ;
3766      <1>      ;
3767      <1>      ;
3768      <1>      ;
3769      <1>      ;
3770      <1>      ;
3771      <1>      ;
3772      <1>      ;
3773      <1>      ;
3774      <1>      ;
3775      <1>      ;
3776      <1>      ;
3777      <1>      ;
3778      <1>      ;
3779      <1>      ;
3780      <1>      ;
3781      <1>      ;
3782      <1>      ;
3783      <1>      ;
3784      <1>      ;
3785      <1>      ;
3786      <1>      ;
3787      <1>      ;
3788      <1>      ;
3789      <1>      ;
3790      <1>      ;
3791      <1>      ;
3792      <1>      ;
3793      <1>      ;
3794      <1>      ;
3795      <1>      ;
3796      <1>      ;
3797      <1>      ;
3798      <1>      ;
3799      <1>      ;
3800      <1>      ;
3801      <1>      ;
3802      <1>      ;
3803      <1>      ;
3804      <1>      ;
3805      <1>      ;
3806      <1>      ;
3807      <1>      ;
3808      <1>      ;
3809      <1>      ;
3810      <1>      ;
3811      <1>      ;
3812      <1>      ;
3813      <1>      ;
3814      <1>      ;
3815      <1>      ;
3816      <1>      ;
3817      <1>      ;
3818      <1>      ;
3819      <1>      ;
3820      <1>      ;
3821      <1>      ;
3822      <1>      ;
3823      <1>      ;
3824      <1>      ;
3825      <1>      ;
3826      <1>      ;
3827      <1>      ;
3828      <1>      ;
3829      <1>      ;
3830      <1>      ;
3831      <1>      ;
3832      <1>      ;
3833      <1>      ;
3834      <1>      ;
3835      <1>      ;
3836      <1>      ;
3837      <1>      ;
3838      <1>      ;
3839      <1>      ;
3840      <1>      ;
3841      <1>      ;
3842      <1>      ;
3843      <1>      ;
3844      <1>      ;
3845      <1>      ;
3846      <1>      ;
3847      <1>      ;
3848      <1>      ;
3849      <1>      ;
3850      <1>      ;
3851      <1>      ;
3852      <1>      ;
3853      <1>      ;
3854      <1>      ;
3855      <1>      ;
3856      <1>      ;
3857      <1>      ;
3858      <1>      ;
3859      <1>      ;
3860      <1>      ;
3861      <1>      ;
3862      <1>      ;
3863      <1>      ;
3864      <1>      ;
3865      <1>      ;
3866      <1>      ;
3867      <1>      ;
3868      <1>      ;
3869      <1>      ;
3870      <1>      ;
3871      <1>      ;
3872      <1>      ;
3873      <1>      ;
3874      <1>      ;
3875      <1>      ;
3876      <1>      ;
3877      <1>      ;
3878      <1>      ;
3879      <1>      ;
3880      <1>      ;
3881      <1>      ;
3882      <1>      ;
3883      <1>      ;
3884      <1>      ;
3885      <1>      ;
3886      <1>      ;
3887      <1>      ;
3888      <1>      ;
3889      <1>      ;
3890      <1>      ;
3891      <1>      ;
3892      <1>      ;
3893      <1>      ;
3894      <1>      ;
3895      <1>      ;
3896      <1>      ;
3897      <1>      ;
3898      <1>      ;
3899      <1>      ;
3900      <1>      ;
3901      <1>      ;
3902      <1>      ;
3903      <1>      ;
3904      <1>      ;
3905      <1>      ;
3906      <1>      ;
3907      <1>      ;
3908      <1>      ;
3909      <1>      ;
3910      <1>      ;
3911      <1>      ;
3912      <1>      ;
3913      <1>      ;
3914      <1>      ;
3915      <1>      ;
3916      <1>      ;
3917      <1>      ;
3918      <1>      ;
3919      <1>      ;
3920      <1>      ;
3921      <1>      ;
3922      <1>      ;
3923      <1>      ;
3924      <1>      ;
3925      <1>      ;
3926      <1>      ;
3927      <1>      ;
3928      <1>      ;
3929      <1>      ;
3930      <1>      ;
3931      <1>      ;
3932      <1>      ;
3933      <1>      ;
3934      <1>      ;
3935      <1>      ;
3936      <1>      ;
3937      <1>      ;
3938      <1>      ;
3939      <1>      ;
3940      <1>      ;
3941      <1>      ;
3942      <1>      ;
3943      <1>      ;
3944      <1>      ;
3945      <1>      ;
3946      <1>      ;
3947      <1>      ;
3948      <1>      ;
3949      <1>      ;
3950      <1>      ;
3951      <1>      ;
3952      <1>      ;
3953      <1>      ;
3954      <1>      ;
3955      <1>      ;
3956      <1>      ;
3957      <1>      ;
3958      <1>      ;
3959      <1>      ;
3960      <1>      ;
3961      <1>      ;
3962      <1>      ;
3963      <1>      ;
3964      <1>      ;
3965      <1>      ;
3966      <1>      ;
3967      <1>      ;
3968      <1>      ;
3969      <1>      ;
3970      <1>      ;
3971      <1>      ;
3972      <1>      ;
3973      <1>      ;
3974      <1>      ;
3975      <1>      ;
3976      <1>      ;
3977      <1>      ;
3978      <1>      ;
3979      <1>      ;
3980      <1>      ;
3981      <1>      ;
3982      <1>      ;
3983      <1>      ;
3984      <1>      ;
3985      <1>      ;
3986      <1>      ;
3987      <1>      ;
3988      <1>      ;
3989      <1>      ;
3990      <1>      ;
3991      <1>      ;
3992      <1>      ;
3993      <1>      ;
3994      <1>      ;
3995      <1>      ;
3996      <1>      ;
3997      <1>      ;
3998      <1>      ;
3999      <1>      ;
4000      <1>      ;
4001      <1>      ;
4002      <1>      ;
4003      <1>      ;
4004      <1>      ;
4005      <1>      ;
4006      <1>      ;
4007      <1>      ;
4008      <1>      ;
4009      <1>      ;
4010      <1>      ;
4011      <1>      ;
4012      <1>      ;
4013      <1>      ;
4014      <1>      ;
4015      <1>      ;
4016      <1>      ;
4017      <1>      ;
4018      <1>      ;
4019      <1>      ;
4020      <1>      ;
4021      <1>      ;
4022      <1>      ;
4023      <1>      ;
4024      <1>      ;
4025      <1>      ;
4026      <1>      ;
4027      <1>      ;
4028      <1>      ;
4029      <1>      ;
4030      <1>      ;
4031      <1>      ;
4032      <1>      ;
4033      <1>      ;
4034      <1>      ;
4035      <1>      ;
4036      <1>      ;
4037      <1>      ;
4038      <1>      ;
4039      <1>      ;
4040      <1>      ;
4041      <1>      ;
4042      <1>      ;
4043      <1>      ;
4044      <1>      ;
4045      <1>      ;
4046      <1>      ;
4047      <1>      ;
4048      <1>      ;
4049      <1>      ;
4050      <1>      ;
4051      <1>      ;
4052      <1>      ;
4053      <1>      ;
4054      <1>      ;
4055      <1>      ;
4056      <1>      ;
4057      <1>      ;
4058      <1>      ;
4059      <1>      ;
4060      <1>      ;
4061      <1>      ;
4062      <1>      ;
4063      <1>      ;
4064      <1>      ;
4065      <1>      ;
4066      <1>      ;
4067      <1>      ;
4068      <1>      ;
4069      <1>      ;
4070      <1>      ;
4071      <1>      ;
4072      <1>      ;
4073      <1>      ;
4074      <1>      ;
4075      <1>      ;
4076      <1>      ;
4077      <1>      ;
4078      <1>      ;
4079      <1>      ;
4080      <1>      ;
4081      <1>      ;
4082      <1>      ;
4083      <1>      ;
4084      <1>      ;
4085      <1>      ;
4086      <1>      ;
4087      <1>      ;
4088      <1>      ;
4
```

```

3419
3420
3421
3422
3423
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
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110

Align 2
;;; 11/03/2015
%include 'kybdata.s' ; KEYBOARD (BIOS) DATA
*****
<1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.5 - kybdata.s
<1> ; -----
<1> ; Last Update: 24/07/2022 (Previous: 17/01/2016)
<1> ; -----
<1> ; Beginning: 17/01/2016
<1> ; -----
<1> ; Assembler: NASM version 2.15 (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> ; 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> ; 24/07/2022 - TRDOS 386 Kernel v2.0.5
<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> ; KEY IDENTIFICATION SCAN TABLES
<1> ; -----
<1> ; ----- TABLES FOR ALT CASE -----
<1> ; ----- ALT-INPUT-TABLE -----
<1> K30: db 82,79,80,81,75 ; 10 NUMBER ON KEYPAD
<1> db 76,77,71,72,73
<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> ; ----- 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 $-_K6
<1> ; ----- MASK_TABLE -----
<1> _K7: db INS_SHIFT ; INSERT MODE SHIFT
<1> db CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
<1> db LEFT_SHIFT,RIGHT_SHIFT
<1> ; ----- TABLES FOR CTRL CASE ----- ; ----- CHARACTERS -----
<1> _K8: db 27,-1,0,-1,-1,-1 ; Esc, 1, 2, 3, 4, 5
<1> db 30,-1,-1,-1,-1,31 ; 6, 7, 8, 9, 0, -
<1> ; db -1,127,-1,17,23,5 ; =, Bksp, Tab, Q, W, E
<1> db -1,127,148,17,23,5 ; 24/07/2022
<1> db 18,20,25,21,9,15 ; R, T, Y, U, I, O
<1> db 16,27,29,10,-1,1 ; P, [, ], Enter, Ctrl, A
<1> db 19,4,6,7,8,10 ; S, D, F, G, H, J
<1> db 11,12,-1,-1,-1,-1 ; K, L, ;, ', LShift
<1> db 28,26,24,3,22,2 ; Bkslash, Z, X, C, V, B
<1> db 14,13,-1,-1,-1,-1 ; N, M, , ., /, RShift
<1> db 150,-1,' ', -1 ; *, ALT, Spc, CL
<1> ; ----- FUNCTIONS -----
<1> db 94,95,96,97,98,99 ; F1 - F6
<1> db 100,101,102,103,-1,-1 ; F7 - F10, NL, SL
<1> db 119,141,132,142,115,143 ; Home, Up, PgUp, -, Left, Pad5
<1> db 116,144,117,145,118,146 ; Right, +, End, Down, PgDn, Ins
<1> db 147,-1,-1,-1,137,138 ; Del, SysReq, Undef, WT, F11, F12
<1> ; ----- TABLES FOR LOWER CASE -----
<1> K10: db 27,'1234567890','=','8,9
<1> db 'qwertyuiop[]',13,-1,'asdfghjkl;',39
<1> db 96,-1,92,'zxcvbnm,./',-1,'*','-1,' ', -1
<1> ; ----- LC TABLE SCAN -----
<1> db 59,60,61,62,63 ; BASE STATE OF F1 - F10
<1> db 64,65,66,67,68
<1> db -1,-1 ; NL, SL
<1> ; ----- KEYPAD TABLE -----
<1> K15: db 71,72,73,-1,75,-1 ; BASE STATE OF KEYPAD KEYS
<1> db 77,-1,79,80,81,82,83
<1> db -1,-1,92,133,134 ; SysRq, Undef, WT, F11, F12
<1> ; ----- TABLES FOR UPPER CASE -----
<1> K11: db 27,'!@#%$',94,'&*()-+',8,0
<1> db 'QWERTYUIOP{}',13,-1,'ASDFGHJKL:'''
<1> db 126,-1,'|ZXCVCNM<>?',-1,'*','-1,' ', -1
<1> ; ----- UC TABLE SCAN -----
<1> K12: db 84,85,86,87,88 ; SHIFTED STATE OF F1 - F10
<1> db 89,90,91,92,93
<1> db -1,-1 ; NL, SL
<1> ; ----- NUM STATE TABLE -----
<1> K14: db '789-456+1230.' ; NUMLOCK STATE OF KEYPAD KEYS
<1> db -1,-1,124,135,136 ; SysRq, Undef, WT, F11, F12
<1> ; -----
<1> ; 26/08/2014
<1> ; Retro UNIX 8086 v1 - UNIX.ASM (03/03/2014)
<1> ; Derived from IBM "pc-at"
<1> ; rombios source code (06/10/1985)
<1> ; 'dseg.inc'
<1> ; -----

```

```

111 <1> ; SYSTEM DATA AREA ;
112 <1> ;-----
113 000067E8 00 <1> BIOS_BREAK db 0 ; BIT 7=1 IF BREAK KEY HAS BEEN PRESSED
114 <1> ;-----
115 <1> ;-----
116 <1> ; KEYBOARD DATA AREAS ;
117 <1> ;-----
118 <1> ;-----
119 000067E9 00 <1> KB_FLAG db 0 ; KEYBOARD SHIFT STATE AND STATUS FLAGS
120 000067EA 00 <1> KB_FLAG_1 db 0 ; SECOND BYTE OF KEYBOARD STATUS
121 000067EB 00 <1> KB_FLAG_2 db 0 ; KEYBOARD LED FLAGS
122 000067EC 00 <1> KB_FLAG_3 db 0 ; KEYBOARD MODE STATE AND TYPE FLAGS
123 000067ED 00 <1> ALT_INPUT db 0 ; STORAGE FOR ALTERNATE KEY PAD ENTRY
124 000067EE [FE670000] <1> BUFFER_START dd KB_BUFFER ; OFFSET OF KEYBOARD BUFFER START
125 000067F2 [1E680000] <1> BUFFER_END dd KB_BUFFER + 32 ; OFFSET OF END OF BUFFER
126 000067F6 [FE670000] <1> BUFFER_HEAD dd KB_BUFFER ; POINTER TO HEAD OF KEYBOARD BUFFER
127 000067FA [FE670000] <1> BUFFER_TAIL dd KB_BUFFER ; POINTER TO TAIL OF KEYBOARD BUFFER
128 <1> ; ----- HEAD = TAIL INDICATES THAT THE BUFFER IS EMPTY
129 000067FE 0000<rep 10h> <1> KB_BUFFER times 16 dw 0 ; ROOM FOR 16 SCAN CODE ENTRIES
130 <1> ;-----
131 <1> ; /// End of KEYBOARD DATA ///
3424 <1> %include 'vidata.s' ; VIDEO (BIOS) DATA
<1> ; *****
<1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.10 - vidata.s
<1> ;-----
<1> ; Last Update: 21/12/2025 (Previous: 17/10/2023 - Kernel v2.0.7)
<1> ;-----
<1> ; Beginning: 16/01/2016
<1> ;-----
<1> ; Assembler: NASM version 2.15 (trdos386.s)
<1> ;-----
<1> ; Turkish Rational DOS
<1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
<1> ;-----
<1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
<1> ; vidata.inc (11/03/2015)
<1> ;-----
<1> ; Derived from 'IBM PC-AT' BIOS source code (1985)
<1> ; *****
<1> ;-----
<1> ; Retro UNIX 386 v1 Kernel - VIDATA.S
<1> ; Last Modification: 11/03/2015
<1> ; (Data section for 'VIDEO.INC')
<1> ;-----
<1> ; ////////// VIDEO DATA //////////
<1> ;-----
<1> ; VIDEO DISPLAY DATA AREA ;
<1> ;-----
28 0000681E 03 <1> CRT_MODE: db 3 ; CURRENT DISPLAY MODE (TYPE)
29 0000681F 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 00006820 50 <1> CRT_COLS: db 80 ; Number of columns
52 <1> ;-----
53 <1> ; 01/07/2016
54 00006821 00 <1> CRT_PALETTE: db 0 ; Current palette setting
55 <1> ;-----
56 <1> ; 03/07/2016
57 00006822 10 <1> CHAR_HEIGHT: db 16 ; Default character height
58 00006823 60 <1> VGA_VIDEO_CTL: db 60h ; ROM BIOS DATA AREA Offset 87h
59 00006824 F9 <1> VGA_SWITCHES: db 0F9h ; Feature Bit Switches (the basic screen)
60 00006825 51 <1> VGA_MODESET_CTL: db 051h ; Basic mode set options (VGA video flags)
61 <1> ; ROM BIOS DATA AREA Offset 89h
62 <1> ; Bit 7, 4 : Mode
63 <1> ; 01 : 400-line mode
64 <1> ; Bit 6: Display switch enabled = 1
65 <1> ; Bit 5: Reserved = 0
66 <1> ; Bit 3: Default palette loading
67 <1> ; disabled = 0
68 <1> ; Bit 2 : Color monitor = 0
69 <1> ; Bit 1 = Gray scale summing
70 <1> ; disabled = 0
71 <1> ; Bit 0 = VGA active = 1
72 00006826 19 <1> VGA_ROWS: db 25
73 <1> ;-----
74 <1> ; 16/01/2016
75 chr_attrib: ; Character color/attributes for video pages (0 to 7)
76 00006827 0707070707070707 <1> db 07h, 07h, 07h, 07h, 07h, 07h, 07h, 07h
77 <1> ; 30/01/2016
78 vmode: <1>
79 0000682F 0303030303030303 <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 00006837 0F0E <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 00006839 90 <1> align 2
89 <1> ;-----
90 <1> vga_modes:
91 <1> ; 25/07/2016
92 <1> ; 09/07/2016
93 <1> ; 03/07/2016
94 <1> ; valid (implemented) video modes (>7, extension to IBM PC CGA modes)
95 0000683A 0302010007040506 <1> db 03h, 02h, 01h, 00h, 07h, 04h, 05h, 06h
96 <1> vga_g_modes: ; 31/07/2016
97 00006842 13F0126A0D0E1011 <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 0000684A [AA680000] <1> dd vga_mode_03h
104 0000684E [AA680000] <1> dd vga_mode_03h ; mode 02h -> mode 03h
105 00006852 [EA680000] <1> dd vga_mode_01h
106 00006856 [EA680000] <1> dd vga_mode_01h ; mode 00h -> mode 01h
107 <1> ;dd vga_mode_07h
108 0000685A [AA680000] <1> dd vga_mode_03h ; mode 07h -> mode 03h
109 0000685E [2A690000] <1> dd vga_mode_04h
110 00006862 [2A690000] <1> dd vga_mode_04h ; mode 05h -> mode 04h
111 00006866 [6A690000] <1> dd vga_mode_06h
112 <1> vga_g_mode_tbl_ptr: ; 21/12/2025
113 0000686A [AA690000] <1> dd vga_mode_13h
114 0000686E [EA690000] <1> dd vga_mode_F0h
115 00006872 [2A6A0000] <1> dd vga_mode_12h
116 00006876 [6A6A0000] <1> dd vga_mode_6Ah
117 0000687A [AA6A0000] <1> dd vga_mode_0Dh
118 0000687E [EA6A0000] <1> dd vga_mode_0Eh
119 00006882 [2A6B0000] <1> dd vga_mode_10h
120 00006886 [6A6B0000] <1> dd vga_mode_11h
121 <1>
122 <1> vga_memmodel:
123 <1> ; 25/07/2016
124 <1> ; 07/07/2016
125 <1> CTEXT equ 0
126 <1> ;MTEXT equ 1
127 <1> MTEXT equ 0 ; mode 07h -> mode 03h
128 <1> CGA equ 2
129 <1> LINEAR8 equ 5
130 <1> PLANAR4 equ 4
131 <1> PLANAR1 equ 3
132 0000688A 00000000000020202 <1> db CTEXT, CTEXT, CTEXT, CTEXT, MTEXT, CGA, CGA, CGA
133 <1> vga_g_memmodel: ; 31/07/2016
134 00006892 0504040404040403 <1> db LINEAR8, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR4, PLANAR1
135 <1> ;vga_pixbits:
136 <1> ; ; 25/07/2016
137 <1> ; ; 08/07/2016
138 <1> ; db 4, 4, 4, 4, 4, 4, 2, 2, 1, 8, 4, 4, 4, 4, 4, 4, 1
139 <1> vga_dac_s:
140 0000689A 020202020001010103- <1> db 2, 2, 2, 2, 0, 1, 1, 1, 3, 3, 2, 2, 1, 1, 2, 2
140 000068A3 03020201010202 <1>
141 <1> ; (vgatables.h, VGAMODES, dac)
142 <1> ; 17/11/2020
143 <1> vga_params:
144 <1> ; 23/11/2020
145 <1> ; 16/11/2020
146 <1> ; 09/11/2020, 10/11/2020, 11/11/2020 (TRDOS 386 v2.0.3)
147 <1> ; 25/07/2016
148 <1> ; 19/07/2016
149 <1> ; 03/07/2016
150 <1> ; derived from 'Plex86/Bochs VGABios' source code
151 <1> ; vgabios-0.7a (2011)
152 <1> ; by the LGPL VGABios Developers Team (2001-2008)
153 <1> ; 'vgatables.h'
154 <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code
155 <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
156 <1>
157 <1> ; 09/11/2020
158 <1> ; Block Structure of Video Parameter Table
159 <1> ;
160 <1> ; Offset # Bytes Contents
161 <1> ;
162 <1> ; 0 1 # columns
163 <1> ; 1 1 # rows - 1
164 <1> ; 2 1 Pixels/character
165 <1> ; 3-4 2 Page length
166 <1> ; 5-8 4 Sequencer Registers
167 <1> ; 9 1 Miscellaneous Register
168 <1> ; 10-34 25 CRTC Registers
169 <1> ; 35-54 20 Attribute Registers
170 <1> ; 55-63 9 Graphics Controller Registers
171 <1> ;
172 <1> ; Ref: Programmer's Guide to EGA, VGA, and Super VGA cards
173 <1> ; (Richard F. Ferraro, 1994)
174 <1>
175 <1> ;
176 <1> vga_mode_03h: ; mode 03h, 80*25 text, CGA colors
177 <1> ; 11/11/2020
178 000068AA 5018100010 <1> db 80, 24, 16, 00h, 10h ; tw, th-1, ch, slength (5)
179 000068AF 00030002 <1> db 00h, 03h, 00h, 02h ; sequ regs (4)
180 000068B3 67 <1> db 67h ; misc reg (1)
181 000068B4 5F4F50825581BF1F <1> db 5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh
182 <1> ; 09/11/2020
183 <1> ;db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
184 000068BC 004F <1> db 00h, 4Fh
185 <1> vga_p_cm_pos equ $ - vga_mode_03h
186 000068BE 0D0E00000000 <1> db 0Dh, 0Eh, 00h, 00h, 00h, 00h
187 000068C4 9C8E8F281F96B9A3 <1> db 9Ch, 8Eh, 8Fh, 28h, 1Fh, 96h, 0B9h, 0A3h
188 000068CC FF <1> db 0FFh ; crtc_regs (25)
189 000068CD 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
190 000068D5 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
191 <1> ; 17/10/2023
192 000068DD 0C000F08 <1> db 0Ch, 00h, 0Fh, 08h ; actl regs (20)
193 <1> ;db 0Ch, 00h, 0Fh, 00h ; 19/11/2020
194 000068E1 0000000000100E0FFF <1> db 00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs (9)
195 <1> ; 09/11/2020
196 <1> ;db 00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 00h, 0FFh ; grdc regs (9)
197 <1> vga_mode_01h: ; mode 01h, 40*25 text, CGA colors
198 000068EA 2818100008 <1> db 40, 24, 16, 00h, 08h ; tw, th-1, ch, slength
199 000068EF 08030002 <1> db 08h, 03h, 00h, 02h ; sequ regs
200 000068F3 67 <1> db 67h ; misc reg
201 000068F4 2D2728902BA0BF1F <1> db 2Dh, 27h, 28h, 90h, 2Bh, 0A0h, 0BFh, 1Fh
202 000068FC 004F0D0E00000000 <1> db 00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h
203 00006904 9C8E8F141F96B9A3 <1> db 9Ch, 8Eh, 8Fh, 14h, 1Fh, 96h, 0B9h, 0A3h
204 0000690C FF <1> db 0FFh ; crtc_regs
205 0000690D 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
206 00006915 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
207 <1> ;db 0Ch, 00h, 0Fh, 08h ; actl regs (20)
208 0000691D 0C000F00 <1> db 0Ch, 00h, 0Fh, 00h ; 19/11/2020
209 00006921 0000000000100E0FFF <1> db 00h, 00h, 00h, 00h, 00h, 10h, 0Eh, 0Fh, 0FFh ; grdc regs
210 <1> ;vga_mode_07h: ; mode 07h, 80*25 text, mono color
211 <1> ; db 80, 24, 16, 00h, 10h ; tw, th-1, ch, slength
212 <1> ; db 00h, 03h, 00h, 02h ; sequ regs
213 <1> ; db 66h ; misc reg
214 <1> ; db 5Fh, 4Fh, 50h, 82h, 55h, 81h, 0BFh, 1Fh
215 <1> ; db 00h, 4Fh, 0Dh, 0Eh, 00h, 00h, 00h, 00h
216 <1> ; db 9Ch, 8Eh, 8Fh, 28h, 0Fh, 96h, 0B9h, 0A3h
217 <1> ; db 0FFh ; crtc_regs
218 <1> ; db 00h, 08h, 08h, 08h, 08h, 08h, 08h, 08h
219 <1> ; db 10h, 18h, 18h, 18h, 18h, 18h, 18h, 18h
220 <1> ; db 0Eh, 00h, 0Fh, 08h ; actl regs
221 <1> ; db 00h, 00h, 00h, 00h, 00h, 10h, 0Ah, 0Fh, 0FFh ; grdc regs
222 <1> vga_mode_04h: ; 320*200 graphics, 4 colors, CGA
223 <1> ; 11/11/2020
224 0000692A 2818080040 <1> db 40, 24, 8, 00h, 40h ; tw, th-1, ch, slength
225 0000692F 09030002 <1> db 09h, 03h, 00h, 02h ; sequ regs

```

```

226 00006933 63 <1> db 63h ; misc reg
227 00006934 2b2728902b80bf1f <1> db 2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh
228 0000693C 00c1000000000000 <1> db 00h, 0c1h, 00h, 00h, 00h, 00h, 00h, 00h
229 00006944 9c8e8f140096b9a2 <1> db 9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0A2h
230 0000694C FF <1> db 0FFh ; crtc_regs
231 0000694D 0013151702040607 <1> db 00h, 13h, 15h, 17h, 02h, 04h, 06h, 07h
232 00006955 1011121314151617 <1> db 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
233 0000695D 01000300 <1> db 01h, 00h, 03h, 00h ; actl regs
234 00006961 0000000000300F0FFF <1> db 00h, 00h, 00h, 00h, 00h, 30h, 0Fh, 0FFh ; grdc regs
235 <1> vga_mode_06h: ; 640*200 graphics, 2 colors, CGA
236 <1> ; 11/11/2020
237 0000696A 5018080040 <1> db 80, 24, 8, 00h, 40h ; tw, th-1, ch, slength
238 0000696F 01010006 <1> db 01h, 01h, 00h, 06h ; sequ regs
239 00006973 63 <1> db 63h ; misc reg
240 00006974 5f4f50825480bf1f <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
241 0000697C 00c1000000000000 <1> db 00h, 0c1h, 00h, 00h, 00h, 00h, 00h, 00h
242 00006984 9c8e8f280096b9c2 <1> db 9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0C2h
243 0000698C FF <1> db 0FFh ; crtc_regs
244 0000698D 0017171717171717 <1> db 00h, 17h, 17h, 17h, 17h, 17h, 17h, 17h
245 00006995 1717171717171717 <1> db 17h, 17h, 17h, 17h, 17h, 17h, 17h, 17h
246 0000699D 01000100 <1> db 01h, 00h, 01, 00h ; actl regs
247 000069A1 0000000000000D0FFF <1> db 00h, 00h, 00h, 00h, 00h, 0Dh, 0Fh, 0FFh ; grdc regs
248 <1> vga_mode_13h: ; mode 13h, 300*200, 256 colors, linear
249 <1> ; 11/11/2020
250 <1> ;db 40, 24, 8, 00h, 20h ; tw, th-1, ch, slength (5)
251 <1> ; 23/11/2020 - 10/11/2020
252 000069AA 28180800FA <1> db 40, 24, 8, 00h, 0FAh ; tw, th-1, ch, slength (5)
253 000069AF 010F000E <1> db 01h, 0Fh, 00h, 0Eh ; sequ regs (4)
254 000069B3 63 <1> db 63h ; misc reg (1)
255 000069B4 5f4f50825480bf1f <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
256 000069BC 0041000000000000 <1> db 00h, 041h, 00h, 00h, 00h, 00h, 00h, 00h
257 000069C4 9c8e8f284096b9a3 <1> db 9Ch, 8Eh, 8Fh, 28h, 40h, 96h, 0B9h, 0A3h
258 <1> ;db 9Ch, 0Eh, 8Fh, 28h, 40h, 96h, 0B9h, 0A3h ; 17/10/2023
259 000069CC FF <1> db 0FFh ; crtc_regs (25)
260 000069CD 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
261 000069D5 08090A0B0C0D0E0F <1> db 08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
262 000069DD 41000F00 <1> db 41h, 00h, 0Fh, 00h ; actl regs (20)
263 <1> ; 10/11/2020
264 <1> ;db 41h, 01h, 0Fh, 13h ; actl regs (20)
265 000069E1 000000000040050FFF <1> db 00h, 00h, 00h, 00h, 00h, 40h, 05h, 0Fh, 0FFh ; grdc regs (9)
266 <1> vga_mode_setl equ $ - vga_mode_13h ; = 64
267 <1> vga_mode_F0h: ; mode X ; 320*240, 256 colors, planar
268 000069EA 2818080000 <1> db 40, 24, 8, 00h, 00h ; tw, th-1, ch, slength
269 000069EF 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
270 000069F3 E3 <1> db 0E3h ; misc reg
271 000069F4 5f4f508254800D3E <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Dh, 3Eh
272 000069FC 0041000000000000 <1> db 00h, 41h, 00h, 00h, 00h, 00h, 00h, 00h
273 00006A04 EAACDF2800E706E3 <1> db 0EAh, 0ACh, 0DFh, 28h, 00h, 0E7h, 06h, 0E3h
274 00006A0C FF <1> db 0FFh ; crtc_regs (25)
275 00006A0D 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
276 00006A15 08090A0B0C0D0E0F <1> db 08h, 09h, 0Ah, 0Bh, 0Ch, 0Dh, 0Eh, 0Fh
277 00006A1D 41000F00 <1> db 41h, 00h, 0Fh, 00h ; actl regs
278 00006A21 000000000040050FFF <1> db 00h, 00h, 00h, 00h, 00h, 40h, 05h, 0Fh, 0FFh ; grdc regs
279 <1> vga_mode_12h: ; mode 12h, 640*480, 16 colors, planar
280 <1> ; 11/11/2020
281 <1> ;db 80, 29, 16, 0, 0 ; tw, th-1, ch, slength
282 <1> ; 09/11/2020
283 00006A2A 501D1000A0 <1> db 80, 29, 16, 00h, 0A0h ; tw, th-1, ch, slength
284 00006A2F 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
285 00006A33 E3 <1> db 0E3h ; misc reg
286 00006A34 5f4f508254800B3E <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh
287 <1> ; 09/11/2020
288 <1> ;db 5Fh, 4Fh, 50h, 82h, 53h, 9Fh, 0Bh, 3Eh
289 00006A3C 0040000000000000 <1> db 00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
290 00006A44 EA8CDF2800E704E3 <1> db 0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h
291 <1> ; 09/11/2020
292 <1> ;db 0E9h, 8Bh, 0DFh, 28h, 00h, 0E7h, 04h, 0E3h
293 00006A4C FF <1> db 0FFh ; crtc_regs
294 00006A4D 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
295 00006A55 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
296 00006A5D 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
297 00006A61 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
298 <1> vga_mode_6Ah: ; mode 6Ah, 800*600, 16 colors, planar
299 <1> ; 11/11/2020
300 00006A6A 6424100000 <1> db 100, 36, 16, 00h, 00h ; tw, th-1, ch, slength
301 00006A6F 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
302 00006A73 E3 <1> db 0E3h ; misc reg
303 00006A74 7f6363836B1B72F0 <1> db 7Fh, 63h, 63h, 83h, 6Bh, 1Bh, 72h, 0F0h
304 00006A7C 0060000000000000 <1> db 00h, 60h, 00h, 00h, 00h, 00h, 00h, 00h
305 00006A84 598D5732005773E3 <1> db 59h, 8Dh, 57h, 32h, 00h, 57h, 73h, 0E3h
306 00006A8C FF <1> db 0FFh ; crtc_regs
307 00006A8D 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
308 00006A95 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
309 00006A9D 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
310 00006AA1 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
311 <1> vga_mode_0Dh: ; mode 0Dh, 320*200, 16 colors, planar
312 00006AAA 2818080020 <1> db 40, 24, 8, 00h, 20h ; tw, th-1, ch, slength
313 00006AAF 090F0006 <1> db 09h, 0Fh, 00h, 06h ; sequ regs
314 00006AB3 63 <1> db 63h ; misc reg
315 00006AB4 2b2728902b80bf1f <1> db 2Dh, 27h, 28h, 90h, 2Bh, 80h, 0BFh, 1Fh
316 00006ABC 00c0000000000000 <1> db 00h, 0c0h, 00h, 00h, 00h, 00h, 00h, 00h
317 00006AC4 9c8e8f140096b9E3 <1> db 9Ch, 8Eh, 8Fh, 14h, 00h, 96h, 0B9h, 0E3h
318 00006ACC FF <1> db 0FFh ; crtc_regs
319 00006ACD 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
320 00006AD5 1011121314151617 <1> db 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
321 00006ADD 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
322 00006AE1 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
323 <1> vga_mode_0Eh: ; mode 0Eh, 640*200, 16 colors, planar
324 00006AEA 5018080040 <1> db 80, 24, 8, 00h, 40h ; tw, th-1, ch, slength
325 00006AEF 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
326 00006AF3 63 <1> db 63h ; misc reg
327 00006AF4 5f4f50825480bf1f <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
328 00006AFC 00c0000000000000 <1> db 00h, 0c0h, 00h, 00h, 00h, 00h, 00h, 00h
329 00006B04 9c8e8f280096b9E3 <1> db 9Ch, 8Eh, 8Fh, 28h, 00h, 96h, 0B9h, 0E3h
330 00006B0C FF <1> db 0FFh ; crtc_regs
331 00006B0D 0001020304050607 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 06h, 07h
332 00006B15 1011121314151617 <1> db 10h, 11h, 12h, 13h, 14h, 15h, 16h, 17h
333 00006B1D 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
334 00006B21 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
335 <1> vga_mode_10h: ; mode 10h, 640*350, 16 colors, planar
336 00006B2A 50180E0080 <1> db 80, 24, 14, 00h, 80h ; tw, th-1, ch, slength
337 00006B2F 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
338 00006B33 A3 <1> db 0A3h ; misc reg
339 00006B34 5f4f50825480bf1f <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0BFh, 1Fh
340 00006B3C 0040000000000000 <1> db 00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
341 00006B44 83855D280F63BAE3 <1> db 83h, 85h, 5Dh, 28h, 0Fh, 63h, 0BAh, 0E3h
342 00006B4C FF <1> db 0FFh ; crtc_regs
343 00006B4D 0001020304051407 <1> db 00h, 01h, 02h, 03h, 04h, 05h, 14h, 07h
344 00006B55 38393A3B3C3D3E3F <1> db 38h, 39h, 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh
345 00006B5D 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
346 00006B61 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
347 <1> vga_mode_11h: ; mode 11h, 640*480, mono color, planar
348 <1> ; 11/11/2020
349 00006B6A 501D1000A0 <1> db 80, 29, 16, 00h, 0A0h ; tw, th-1, ch, slength

```



```

350 00006B6F 010F0006 <1> db 01h, 0Fh, 00h, 06h ; sequ regs
351 00006B73 E3 <1> db 0E3h ; misc reg
352 00006B74 5F4F508254800B3E <1> db 5Fh, 4Fh, 50h, 82h, 54h, 80h, 0Bh, 3Eh
353 00006B7C 0040000000000000 <1> db 00h, 40h, 00h, 00h, 00h, 00h, 00h, 00h
354 00006B84 EA8CDF2800E704C3 <1> db 0EAh, 8Ch, 0DFh, 28h, 00h, 0E7h, 04h, 0C3h ; 11/11/2020
355 00006B8C FF <1> db 0FFh ; crtc regs
356 00006B8D 003F003F003F003F <1> db 00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh
357 00006B95 003F003F003F003F <1> db 00h, 3Fh, 00h, 3Fh, 00h, 3Fh, 00h, 3Fh
358 00006B9D 01000F00 <1> db 01h, 00h, 0Fh, 00h ; actl regs
359 00006BA1 000000000000050FFF <1> db 00h, 00h, 00h, 00h, 00h, 00h, 05h, 0Fh, 0FFh ; grdc regs
360 <1> end_of_vga_params:
361 <1>
362 <1> ; /// End Of VIDEO DATA ///
363 <1>
364 <1> ; 23/11/2020
365 <1> ; VBE 2 BOCHS/QEMU emulator extensions
366 <1> ; for TRDOS 386 v2 kernel (video bios)
367 <1>
368 <1> ; vbetables.h by Volker Rupper (02/01/2020)
369 <1>
370 <1> b_vbe_modes:
371 <1> ;/* standard VESA modes */
372 00006BAA 0001800290010800 <1> dw 100h, 640, 400, 8
373 00006BB2 01018002E0010800 <1> dw 101h, 640, 480, 8
374 00006BBA 0301200358020800 <1> dw 103h, 800, 600, 8
375 00006BC2 0501000400030800 <1> dw 105h, 1024, 768, 8
376 00006BCA 0E014001C8001000 <1> dw 10Eh, 320, 200, 16
377 00006BD2 0F014001C8001800 <1> dw 10Fh, 320, 200, 24
378 00006BDA 11018002E0011000 <1> dw 111h, 640, 480, 16
379 00006BE2 12018002E0011800 <1> dw 112h, 640, 480, 24
380 00006BEA 1401200358021000 <1> dw 114h, 800, 600, 16
381 00006BF2 1501200358021800 <1> dw 115h, 800, 600, 24
382 00006BFA 1701000400031000 <1> dw 117h, 1024, 768, 16
383 00006C02 1801000400031800 <1> dw 118h, 1024, 768, 24
384 <1>
385 <1> ;/* BOCHS/PLEX86 'own' mode numbers */
386 00006C0A 40014001C8002000 <1> dw 140h, 320, 200, 32
387 00006C12 4101800290012000 <1> dw 141h, 640, 400, 32
388 00006C1A 42018002E0012000 <1> dw 142h, 640, 480, 32
389 00006C22 4301200358022000 <1> dw 143h, 800, 600, 32
390 00006C2A 4401000400032000 <1> dw 144h, 1024, 768, 32
391 00006C32 46014001C8000800 <1> dw 146h, 320, 200, 8
392 00006C3A 8D010005D0021000 <1> dw 18Dh, 1280, 720, 16
393 00006C42 8E010005D0021800 <1> dw 18Eh, 1280, 720, 24
394 00006C4A 8F010005D0022000 <1> dw 18Fh, 1280, 720, 32
395 00006C52 9001800738041000 <1> dw 190h, 1920, 1080, 16
396 00006C5A 9101800738041800 <1> dw 191h, 1920, 1080, 24
397 00006C62 9201800738042000 <1> dw 192h, 1920, 1080, 32
398 <1>
399 <1> end_of_b_vbe_modes:
400 <1>
401 <1> MA1 equ VBE_MODE_ATTRIBUTE_SUPPORTED
402 <1> MA2 equ VBE_MODE_ATTRIBUTE_EXTENDED_INFO_AVAILABLE
403 <1> MA3 equ VBE_MODE_ATTRIBUTE_COLOR_MODE
404 <1> MA4 equ VBE_MODE_ATTRIBUTE_LINEAR_FRAME_BUFFER_MODE
405 <1> MA5 equ VBE_MODE_ATTRIBUTE_GRAPHICS_MODE
406 <1>
407 <1> MODE_ATTRIBUTES equ MA1|MA2|MA3|MA4|MA5
408 <1>
409 <1> WA1 equ VBE_WINDOW_ATTRIBUTE_RELOCATABLE
410 <1> WA2 equ VBE_WINDOW_ATTRIBUTE_READABLE
411 <1> WA3 equ VBE_WINDOW_ATTRIBUTE_WRITEABLE
412 <1>
413 <1> WINA_ATTRIBUTES equ WA1|WA2|WA3
414 <1>
415 <1> ; 24/11/2020
416 <1>
417 <1> %if 0
418 <1>
419 <1> MODE_INFO_LIST:
420 <1>
421 <1> ; 24/11/2020
422 <1> ; '%if 0' disables 24 mode info tables here, until %endif
423 <1> ; ('set_mode_info_list' will set only 1 list for selected mode)
424 <1> ; (Purpose: To save about 1 KB kernel size by removing fixed data)
425 <1>
426 <1> dw 0100h ; 640x400x8
427 <1> ModeAttributes1: dw MODE_ATTRIBUTES
428 <1> WinAttributes1: dw WINA_ATTRIBUTES
429 <1> WinBAttributes1: db 0
430 <1> WinGranularity1: dw VBE_DISPI_BANK_SIZE_KB
431 <1> WinSize1: dw VBE_DISPI_BANK_SIZE_KB
432 <1> WinASegment1: dw VGAMEM_GRAPH
433 <1> WinBSegment1: dw 0000h
434 <1> WinFuncPtr1: dd 0
435 <1> BytesPerScanLine1: dw 640
436 <1> XResolution1: dw 640
437 <1> YResolution1: dw 400
438 <1> XCharSize1: db 8
439 <1> YCharSize1: db 16
440 <1> NumberOfPlanes1: db 1
441 <1> BitsPerPixel1: db 8
442 <1> NumberOfBanks1: db 4
443 <1> MemoryModel1: db VBE_MEMORYMODEL_PACKED_PIXEL
444 <1> BankSize1: db 0
445 <1> NumberOfImagePages1: db 64
446 <1> Reserved_page1: db 0
447 <1> RedMaskSize1: db 0
448 <1> RedFieldPosition1: db 0
449 <1> GreenMaskSize1: db 0
450 <1> GreenFieldPosition1: db 0
451 <1> BlueMaskSize1: db 0
452 <1> BlueFieldPosition1: db 0
453 <1> RsvdMaskSize1: db 0
454 <1> RsvdFieldPosition1: db 0
455 <1> DirectColorModeInfo1: db 0
456 <1> PhysBasePtr1: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
457 <1> OffScreenMemOffset1: dd 0
458 <1> OffScreenMemSize1: dw 0
459 <1> LinBytesPerScanLine1: dw 640
460 <1> BnkNumberOfPages1: db 0
461 <1> LinNumberOfPages1: db 0
462 <1> LinRedMaskSize1: db 0
463 <1> LinRedFieldPosition1: db 0
464 <1> LinGreenMaskSize1: db 0
465 <1> LinGreenFieldPosition1: db 0
466 <1> LinBlueMaskSize1: db 0
467 <1> LinBlueFieldPosition1: db 0
468 <1> LinRsvdMaskSize1: db 0
469 <1> LinRsvdFieldPosition1: db 0
470 <1> MaxPixelClock1: dd 0
471 <1>
472 <1> dw 0101h ; 640x480x8
473 <1> ModeAttributes2: dw MODE_ATTRIBUTES

```

```

474 <1> WinAttributes2: db WINA_ATTRIBUTES
475 <1> WinAttributes2: db 0
476 <1> WinGranularity2: dw VBE_DISPI_BANK_SIZE_KB
477 <1> WinSize2: dw VBE_DISPI_BANK_SIZE_KB
478 <1> WinASegment2: dw VGAMEM_GRAPH
479 <1> WinBSegment2: dw 0000h
480 <1> WinFuncPtr2: dd 0
481 <1> BytesPerScanLine2: dw 640
482 <1> XResolution2: dw 640
483 <1> YResolution2: dw 480
484 <1> XCharSize2: db 8
485 <1> YCharSize2: db 16
486 <1> NumberOfPlanes2: db 1
487 <1> BitsPerPixel2: db 8
488 <1> NumberOfBanks2: db 5
489 <1> MemoryModel2: db VBE_MEMORYMODEL_PACKED_PIXEL
490 <1> BankSize2: db 0
491 <1> NumberOfImagePages2: db 53
492 <1> Reserved_page2: db 0
493 <1> RedMaskSize2: db 0
494 <1> RedFieldPosition2: db 0
495 <1> GreenMaskSize2: db 0
496 <1> GreenFieldPosition2: db 0
497 <1> BlueMaskSize2: db 0
498 <1> BlueFieldPosition2: db 0
499 <1> RsvdMaskSize2: db 0
500 <1> RsvdFieldPosition2: db 0
501 <1> DirectColorModeInfo2: db 0
502 <1> PhysBasePtr2: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
503 <1> OffScreenMemOffset2: dd 0
504 <1> OffScreenMemSize2: dw 0
505 <1> LinBytesPerScanLine2: dw 640
506 <1> BnkNumberOfPages2: db 0
507 <1> LinNumberOfPages2: db 0
508 <1> LinRedMaskSize2: db 0
509 <1> LinRedFieldPosition2: db 0
510 <1> LinGreenMaskSize2: db 0
511 <1> LinGreenFieldPosition2: db 0
512 <1> LinBlueMaskSize2: db 0
513 <1> LinBlueFieldPosition2: db 0
514 <1> LinRsvdMaskSize2: db 0
515 <1> LinRsvdFieldPosition2: db 0
516 <1> MaxPixelClock2: dd 0
517 <1>
518 <1> ModeAttributes3: dw 0103h ; 800x600x8
519 <1> WinAttributes3: dw MODE_ATTRIBUTES
520 <1> WinAttributes3: db WINA_ATTRIBUTES
521 <1> WinAttributes3: db 0
522 <1> WinGranularity3: dw VBE_DISPI_BANK_SIZE_KB
523 <1> WinSize3: dw VBE_DISPI_BANK_SIZE_KB
524 <1> WinASegment3: dw VGAMEM_GRAPH
525 <1> WinBSegment3: dw 0000h
526 <1> WinFuncPtr3: dd 0
527 <1> BytesPerScanLine3: dw 800
528 <1> XResolution3: dw 800
529 <1> YResolution3: dw 600
530 <1> XCharSize3: db 8
531 <1> YCharSize3: db 16
532 <1> NumberOfPlanes3: db 1
533 <1> BitsPerPixel3: db 8
534 <1> NumberOfBanks3: db 8
535 <1> MemoryModel3: db VBE_MEMORYMODEL_PACKED_PIXEL
536 <1> BankSize3: db 0
537 <1> NumberOfImagePages3: db 33
538 <1> Reserved_page3: db 0
539 <1> RedMaskSize3: db 0
540 <1> RedFieldPosition3: db 0
541 <1> GreenMaskSize3: db 0
542 <1> GreenFieldPosition3: db 0
543 <1> BlueMaskSize3: db 0
544 <1> BlueFieldPosition3: db 0
545 <1> RsvdMaskSize3: db 0
546 <1> RsvdFieldPosition3: db 0
547 <1> DirectColorModeInfo3: db 0
548 <1> PhysBasePtr3: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
549 <1> OffScreenMemOffset3: dd 0
550 <1> OffScreenMemSize3: dw 0
551 <1> LinBytesPerScanLine3: dw 800
552 <1> BnkNumberOfPages3: db 0
553 <1> LinNumberOfPages3: db 0
554 <1> LinRedMaskSize3: db 0
555 <1> LinRedFieldPosition3: db 0
556 <1> LinGreenMaskSize3: db 0
557 <1> LinGreenFieldPosition3: db 0
558 <1> LinBlueMaskSize3: db 0
559 <1> LinBlueFieldPosition3: db 0
560 <1> LinRsvdMaskSize3: db 0
561 <1> LinRsvdFieldPosition3: db 0
562 <1> MaxPixelClock3: dd 0
563 <1>
564 <1> ModeAttributes4: dw 0105h ; 1024x768x8
565 <1> WinAttributes4: dw MODE_ATTRIBUTES
566 <1> WinAttributes4: db WINA_ATTRIBUTES
567 <1> WinAttributes4: db 0
568 <1> WinGranularity4: dw VBE_DISPI_BANK_SIZE_KB
569 <1> WinSize4: dw VBE_DISPI_BANK_SIZE_KB
570 <1> WinASegment4: dw VGAMEM_GRAPH
571 <1> WinBSegment4: dw 0000h
572 <1> WinFuncPtr4: dd 0
573 <1> BytesPerScanLine4: dw 1024
574 <1> XResolution4: dw 1024
575 <1> YResolution4: dw 768
576 <1> XCharSize4: db 8
577 <1> YCharSize4: db 16
578 <1> NumberOfPlanes4: db 1
579 <1> BitsPerPixel4: db 8
580 <1> NumberOfBanks4: db 12
581 <1> MemoryModel4: db VBE_MEMORYMODEL_PACKED_PIXEL
582 <1> BankSize4: db 0
583 <1> NumberOfImagePages4: db 20
584 <1> Reserved_page4: db 0
585 <1> RedMaskSize4: db 0
586 <1> RedFieldPosition4: db 0
587 <1> GreenMaskSize4: db 0
588 <1> GreenFieldPosition4: db 0
589 <1> BlueMaskSize4: db 0
590 <1> BlueFieldPosition4: db 0
591 <1> RsvdMaskSize4: db 0
592 <1> RsvdFieldPosition4: db 0
593 <1> DirectColorModeInfo4: db 0
594 <1> PhysBasePtr4: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
595 <1> OffScreenMemOffset4: dd 0
596 <1> OffScreenMemSize4: dw 0
597 <1> LinBytesPerScanLine4: dw 1024

```

```

598 <1> BnkNumberOfPages4:      db      0
599 <1> LinNumberOfPages4:      db      0
600 <1> LinRedMaskSize4:        db      0
601 <1> LinRedFieldPosition4:    db      0
602 <1> LinGreenMaskSize4:       db      0
603 <1> LinGreenFieldPosition4:  db      0
604 <1> LinBlueMaskSize4:        db      0
605 <1> LinBlueFieldPosition4:   db      0
606 <1> LinRsvdMaskSize4:        db      0
607 <1> LinRsvdFieldPosition4:   db      0
608 <1> MaxPixelClock4:         dd      0
609 <1>
610 <1>
611 <1> ModeAttributes5:        dw      010Eh ; 320x200x16
612 <1> WinAAttributes5:        dw      MODE_ATTRIBUTES
613 <1> WinBAttributes5:        db      0
614 <1> WinGranularity5:         dw      VBE_DISPI_BANK_SIZE_KB
615 <1> WinSize5:               dw      VBE_DISPI_BANK_SIZE_KB
616 <1> WinASegment5:           dw      VGAMEM_GRAPH
617 <1> WinBSegment5:           dw      0000h
618 <1> WinFuncPtr5:            dd      0
619 <1> BytesPerScanLine5:      dw      640
620 <1> XResolution5:           dw      320
621 <1> YResolution5:           dw      200
622 <1> XCharSize5:             db      8
623 <1> YCharSize5:             db      16
624 <1> NumberOfPlanes5:        db      1
625 <1> BitsPerPixel5:          db      16
626 <1> NumberOfBanks5:         db      2
627 <1> MemoryModel5:          db      VBE_MEMORYMODEL_DIRECT_COLOR
628 <1> BankSize5:              db      0
629 <1> NumberOfImagePages5:    db      130
630 <1> Reserved_page5:         db      0
631 <1> RedMaskSize5:           db      5
632 <1> RedFieldPosition5:      db      11
633 <1> GreenMaskSize5:         db      6
634 <1> GreenFieldPosition5:    db      5
635 <1> BlueMaskSize5:          db      5
636 <1> BlueFieldPosition5:     db      0
637 <1> RsvdMaskSize5:          db      0
638 <1> RsvdFieldPosition5:     db      0
639 <1> DirectColorModeInfo5:   db      0
640 <1> PhysBasePtr5:           dd      VBE_DISPI_LFB_PHYSICAL_ADDRESS
641 <1> OffScreenMemOffset5:    dd      0
642 <1> OffScreenMemSize5:      dw      0
643 <1> LinBytesPerScanLine5:   dw      640
644 <1> BnkNumberOfPages5:      db      0
645 <1> LinNumberOfPages5:      db      0
646 <1> LinRedMaskSize5:        db      5
647 <1> LinRedFieldPosition5:   db      11
648 <1> LinGreenMaskSize5:      db      6
649 <1> LinGreenFieldPosition5: db      5
650 <1> LinBlueMaskSize5:       db      5
651 <1> LinBlueFieldPosition5:  db      0
652 <1> LinRsvdMaskSize5:       db      0
653 <1> LinRsvdFieldPosition5:  db      0
654 <1> MaxPixelClock5:         dd      0
655 <1>
656 <1>
657 <1> ModeAttributes6:        dw      010Fh ; 320x200x24
658 <1> WinAAttributes6:        dw      MODE_ATTRIBUTES
659 <1> WinBAttributes6:        db      0
660 <1> WinGranularity6:         dw      VBE_DISPI_BANK_SIZE_KB
661 <1> WinSize6:               dw      VBE_DISPI_BANK_SIZE_KB
662 <1> WinASegment6:           dw      VGAMEM_GRAPH
663 <1> WinBSegment6:           dw      0000h
664 <1> WinFuncPtr6:            dd      0
665 <1> BytesPerScanLine6:      dw      960
666 <1> XResolution6:           dw      320
667 <1> YResolution6:           dw      200
668 <1> XCharSize6:             db      8
669 <1> YCharSize6:             db      16
670 <1> NumberOfPlanes6:        db      1
671 <1> BitsPerPixel6:          db      24
672 <1> NumberOfBanks6:         db      3
673 <1> MemoryModel6:          db      VBE_MEMORYMODEL_DIRECT_COLOR
674 <1> BankSize6:              db      0
675 <1> NumberOfImagePages6:    db      86
676 <1> Reserved_page6:         db      0
677 <1> RedMaskSize6:           db      8
678 <1> RedFieldPosition6:      db      16
679 <1> GreenMaskSize6:         db      8
680 <1> GreenFieldPosition6:    db      8
681 <1> BlueMaskSize6:          db      8
682 <1> BlueFieldPosition6:     db      0
683 <1> RsvdMaskSize6:          db      0
684 <1> RsvdFieldPosition6:     db      0
685 <1> DirectColorModeInfo6:   db      0
686 <1> PhysBasePtr6:           dd      VBE_DISPI_LFB_PHYSICAL_ADDRESS
687 <1> OffScreenMemOffset6:    dd      0
688 <1> OffScreenMemSize6:      dw      0
689 <1> LinBytesPerScanLine6:   dw      960
690 <1> BnkNumberOfPages6:      db      0
691 <1> LinNumberOfPages6:      db      0
692 <1> LinRedMaskSize6:        db      8
693 <1> LinRedFieldPosition6:   db      16
694 <1> LinGreenMaskSize6:      db      8
695 <1> LinGreenFieldPosition6: db      8
696 <1> LinBlueMaskSize6:       db      8
697 <1> LinBlueFieldPosition6:  db      0
698 <1> LinRsvdMaskSize6:       db      0
699 <1> LinRsvdFieldPosition6:  db      0
700 <1> MaxPixelClock6:         dd      0
701 <1>
702 <1>
703 <1> ModeAttributes7:        dw      0111h ; 640x480x16
704 <1> WinAAttributes7:        dw      MODE_ATTRIBUTES
705 <1> WinBAttributes7:        db      0
706 <1> WinGranularity7:         dw      VBE_DISPI_BANK_SIZE_KB
707 <1> WinSize7:               dw      VBE_DISPI_BANK_SIZE_KB
708 <1> WinASegment7:           dw      VGAMEM_GRAPH
709 <1> WinBSegment7:           dw      0000h
710 <1> WinFuncPtr7:            dd      0
711 <1> BytesPerScanLine7:      dw      1280
712 <1> XResolution7:           dw      640
713 <1> YResolution7:           dw      480
714 <1> XCharSize7:             db      8
715 <1> YCharSize7:             db      16
716 <1> NumberOfPlanes7:        db      1
717 <1> BitsPerPixel7:          db      16
718 <1> NumberOfBanks7:         db      10
719 <1> MemoryModel7:          db      VBE_MEMORYMODEL_DIRECT_COLOR
720 <1> BankSize7:              db      0
721 <1> NumberOfImagePages7:    db      26

```

```

722 <1> Reserved_page7: db 0
723 <1> RedMaskSize7: db 5
724 <1> RedFieldPosition7: db 11
725 <1> GreenMaskSize7: db 6
726 <1> GreenFieldPosition7: db 5
727 <1> BlueMaskSize7: db 5
728 <1> BlueFieldPosition7: db 0
729 <1> RsvdMaskSize7: db 0
730 <1> RsvdFieldPosition7: db 0
731 <1> DirectColorModeInfo7: db 0
732 <1> PhysBasePtr7: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS,
733 <1> OffScreenMemOffset7: dd 0
734 <1> OffScreenMemSize7: dw 0
735 <1> LinBytesPerScanLine7: dw 1280
736 <1> BnkNumberOfPages7: db 0
737 <1> LinNumberOfPages7: db 0
738 <1> LinRedMaskSize7: db 5
739 <1> LinRedFieldPosition7: db 11
740 <1> LinGreenMaskSize7: db 6
741 <1> LinGreenFieldPosition7: db 5
742 <1> LinBlueMaskSize7: db 5
743 <1> LinBlueFieldPosition7: db 0
744 <1> LinRsvdMaskSize7: db 0
745 <1> LinRsvdFieldPosition7: db 0
746 <1> MaxPixelClock7: dd 0
747 <1>
748 <1>
749 <1> ModeAttributes8: dw 0112h ; 640x480x24
750 <1> WinAAttributes8: dw MODE_ATTRIBUTES
751 <1> WinBAttributes8: db WINA_ATTRIBUTES
752 <1> WinGranularity8: dw VBE_DISPI_BANK_SIZE_KB
753 <1> WinSize8: dw VBE_DISPI_BANK_SIZE_KB
754 <1> WinASegment8: dw VGAMEM_GRAPH
755 <1> WinBSegment8: dw 0000h
756 <1> WinFuncPtr8: dd 0
757 <1> BytesPerScanLine8: dw 1920
758 <1> XResolution8: dw 640
759 <1> YResolution8: dw 480
760 <1> XCharSize8: db 8
761 <1> YCharSize8: db 16
762 <1> NumberOfPlanes8: db 1
763 <1> BitsPerPixel8: db 24
764 <1> NumberOfBanks8: db 15
765 <1> MemoryModel8: db VBE_MEMORYMODEL_DIRECT_COLOR
766 <1> BankSize8: db 0
767 <1> NumberOfImagePages8: db 17
768 <1> Reserved_page8: db 0
769 <1> RedMaskSize8: db 8
770 <1> RedFieldPosition8: db 16
771 <1> GreenMaskSize8: db 8
772 <1> GreenFieldPosition8: db 8
773 <1> BlueMaskSize8: db 8
774 <1> BlueFieldPosition8: db 0
775 <1> RsvdMaskSize8: db 0
776 <1> RsvdFieldPosition8: db 0
777 <1> DirectColorModeInfo8: db 0
778 <1> PhysBasePtr8: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
779 <1> OffScreenMemOffset8: dd 0
780 <1> OffScreenMemSize8: dw 0
781 <1> LinBytesPerScanLine8: dw 1920
782 <1> BnkNumberOfPages8: db 0
783 <1> LinNumberOfPages8: db 0
784 <1> LinRedMaskSize8: db 8
785 <1> LinRedFieldPosition8: db 16
786 <1> LinGreenMaskSize8: db 8
787 <1> LinGreenFieldPosition8: db 8
788 <1> LinBlueMaskSize8: db 8
789 <1> LinBlueFieldPosition8: db 0
790 <1> LinRsvdMaskSize8: db 0
791 <1> LinRsvdFieldPosition8: db 0
792 <1> MaxPixelClock8: dd 0
793 <1>
794 <1>
795 <1> ModeAttributes9: dw 0114h ; 800x600x16
796 <1> WinAAttributes9: dw MODE_ATTRIBUTES
797 <1> WinBAttributes9: db WINA_ATTRIBUTES
798 <1> WinGranularity9: dw VBE_DISPI_BANK_SIZE_KB
799 <1> WinSize9: dw VBE_DISPI_BANK_SIZE_KB
800 <1> WinASegment9: dw VGAMEM_GRAPH
801 <1> WinBSegment9: dw 0000h
802 <1> WinFuncPtr9: dd 0
803 <1> BytesPerScanLine9: dw 1600
804 <1> XResolution9: dw 800
805 <1> YResolution9: dw 600
806 <1> XCharSize9: db 8
807 <1> YCharSize9: db 16
808 <1> NumberOfPlanes9: db 1
809 <1> BitsPerPixel9: db 16
810 <1> NumberOfBanks9: db 15
811 <1> MemoryModel9: db VBE_MEMORYMODEL_DIRECT_COLOR
812 <1> BankSize9: db 0
813 <1> NumberOfImagePages9: db 16
814 <1> Reserved_page9: db 0
815 <1> RedMaskSize9: db 5
816 <1> RedFieldPosition9: db 11
817 <1> GreenMaskSize9: db 6
818 <1> GreenFieldPosition9: db 5
819 <1> BlueMaskSize9: db 5
820 <1> BlueFieldPosition9: db 0
821 <1> RsvdMaskSize9: db 0
822 <1> RsvdFieldPosition9: db 0
823 <1> DirectColorModeInfo9: db 0
824 <1> PhysBasePtr9: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
825 <1> OffScreenMemOffset9: dd 0
826 <1> OffScreenMemSize9: dw 0
827 <1> LinBytesPerScanLine9: dw 1600
828 <1> BnkNumberOfPages9: db 0
829 <1> LinNumberOfPages9: db 0
830 <1> LinRedMaskSize9: db 5
831 <1> LinRedFieldPosition9: db 11
832 <1> LinGreenMaskSize9: db 6
833 <1> LinGreenFieldPosition9: db 5
834 <1> LinBlueMaskSize9: db 5
835 <1> LinBlueFieldPosition9: db 0
836 <1> LinRsvdMaskSize9: db 0
837 <1> LinRsvdFieldPosition9: db 0
838 <1> MaxPixelClock9: dd 0
839 <1>
840 <1>
841 <1> ModeAttributes10: dw 0115h ; 800x600x24
842 <1> WinAAttributes10: dw MODE_ATTRIBUTES
843 <1> WinBAttributes10: db WINA_ATTRIBUTES
844 <1> WinGranularity10: dw VBE_DISPI_BANK_SIZE_KB
845 <1> WinSize10: dw VBE_DISPI_BANK_SIZE_KB

```

```

846 <1> WinASegment10: dw VGAMEM_GRAPH
847 <1> WinBSegment10: dw 0000h
848 <1> WinFuncPtr10: dd 0
849 <1> BytesPerScanLine10: dw 2400
850 <1> XResolution10: dw 800
851 <1> YResolution10: dw 600
852 <1> XCharSize10: db 8
853 <1> YCharSize10: db 16
854 <1> NumberOfPlanes10: db 1
855 <1> BitsPerPixel10: db 24
856 <1> NumberOfBanks10: db 22
857 <1> MemoryModel10: db VBE_MEMORYMODEL_DIRECT_COLOR
858 <1> BankSize10: db 0
859 <1> NumberOfImagePages10: db 10
860 <1> Reserved_page10: db 0
861 <1> RedMaskSize10: db 8
862 <1> RedFieldPosition10: db 16
863 <1> GreenMaskSize10: db 8
864 <1> GreenFieldPosition10: db 8
865 <1> BlueMaskSize10: db 8
866 <1> BlueFieldPosition10: db 0
867 <1> RsvdMaskSize10: db 0
868 <1> RsvdFieldPosition10: db 0
869 <1> DirectColorModeInfo10: db 0
870 <1> PhysBasePtr10: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
871 <1> OffScreenMemOffset10: dd 0
872 <1> OffScreenMemSize10: dw 0
873 <1> LinBytesPerScanLine10: dw 2400
874 <1> BnkNumberOfPages10: db 0
875 <1> LinNumberOfPages10: db 0
876 <1> LinRedMaskSize10: db 8
877 <1> LinRedFieldPosition10: db 16
878 <1> LinGreenMaskSize10: db 8
879 <1> LinGreenFieldPosition10: db 8
880 <1> LinBlueMaskSize10: db 8
881 <1> LinBlueFieldPosition10: db 0
882 <1> LinRsvdMaskSize10: db 0
883 <1> LinRsvdFieldPosition10: db 0
884 <1> MaxPixelClock10: dd 0
885 <1>
886 <1>
887 <1> ModeAttributes11: dw 0117h ; 1024x768x16
888 <1> WinAAttributes11: dw MODE_ATTRIBUTES
889 <1> WinBAttributes11: db WINA_ATTRIBUTES
890 <1> WinGranularity11: dw VBE_DISPI_BANK_SIZE_KB
891 <1> WinSize11: dw VBE_DISPI_BANK_SIZE_KB
892 <1> WinASegment11: dw VGAMEM_GRAPH
893 <1> WinBSegment11: dw 0000h
894 <1> WinFuncPtr11: dd 0
895 <1> BytesPerScanLine11: dw 2048
896 <1> XResolution11: dw 1024
897 <1> YResolution11: dw 768
898 <1> XCharSize11: db 8
899 <1> YCharSize11: db 16
900 <1> NumberOfPlanes11: db 1
901 <1> BitsPerPixel11: db 16
902 <1> NumberOfBanks11: db 24
903 <1> MemoryModel11: db VBE_MEMORYMODEL_DIRECT_COLOR
904 <1> BankSize11: db 0
905 <1> NumberOfImagePages11: db 9
906 <1> Reserved_page11: db 0
907 <1> RedMaskSize11: db 5
908 <1> RedFieldPosition11: db 11
909 <1> GreenMaskSize11: db 6
910 <1> GreenFieldPosition11: db 5
911 <1> BlueMaskSize11: db 5
912 <1> BlueFieldPosition11: db 0
913 <1> RsvdMaskSize11: db 0
914 <1> RsvdFieldPosition11: db 0
915 <1> DirectColorModeInfo11: db 0
916 <1> PhysBasePtr11: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS,
917 <1> OffScreenMemOffset11: dd 0
918 <1> OffScreenMemSize11: dw 0
919 <1> LinBytesPerScanLine11: dw 2048
920 <1> BnkNumberOfPages11: db 0
921 <1> LinNumberOfPages11: db 0
922 <1> LinRedMaskSize11: db 5
923 <1> LinRedFieldPosition11: db 11
924 <1> LinGreenMaskSize11: db 6
925 <1> LinGreenFieldPosition11: db 5
926 <1> LinBlueMaskSize11: db 5
927 <1> LinBlueFieldPosition11: db 0
928 <1> LinRsvdMaskSize11: db 0
929 <1> LinRsvdFieldPosition11: db 0
930 <1> MaxPixelClock11: dd 0
931 <1>
932 <1>
933 <1> ModeAttributes12: dw 0118h ; 1024x768x24
934 <1> WinAAttributes12: dw MODE_ATTRIBUTES
935 <1> WinBAttributes12: db WINA_ATTRIBUTES
936 <1> WinGranularity12: dw VBE_DISPI_BANK_SIZE_KB
937 <1> WinSize12: dw VBE_DISPI_BANK_SIZE_KB
938 <1> WinASegment12: dw VGAMEM_GRAPH
939 <1> WinBSegment12: dw 0000h
940 <1> WinFuncPtr12: dd 0
941 <1> BytesPerScanLine12: dw 3072
942 <1> XResolution12: dw 1024
943 <1> YResolution12: dw 768
944 <1> XCharSize12: db 8
945 <1> YCharSize12: db 16
946 <1> NumberOfPlanes12: db 1
947 <1> BitsPerPixel12: db 24
948 <1> NumberOfBanks12: db 36
949 <1> MemoryModel12: db VBE_MEMORYMODEL_DIRECT_COLOR
950 <1> BankSize12: db 0
951 <1> NumberOfImagePages12: db 6
952 <1> Reserved_page12: db 0
953 <1> RedMaskSize12: db 8
954 <1> RedFieldPosition12: db 16
955 <1> GreenMaskSize12: db 8
956 <1> GreenFieldPosition12: db 8
957 <1> BlueMaskSize12: db 8
958 <1> BlueFieldPosition12: db 0
959 <1> RsvdMaskSize12: db 0
960 <1> RsvdFieldPosition12: db 0
961 <1> DirectColorModeInfo12: db 0
962 <1> PhysBasePtr12: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
963 <1> OffScreenMemOffset12: dd 0
964 <1> OffScreenMemSize12: dw 0
965 <1> LinBytesPerScanLine12: dw 3072
966 <1> BnkNumberOfPages12: db 0
967 <1> LinNumberOfPages12: db 0
968 <1> LinRedMaskSize12: db 8
969 <1> LinRedFieldPosition12: db 16

```

```

970 <1> LinGreenMaskSize12: db 8
971 <1> LinGreenFieldPosition12: db 8
972 <1> LinBlueMaskSize12: db 8
973 <1> LinBlueFieldPosition12: db 0
974 <1> LinRsvdMaskSize12: db 0
975 <1> LinRsvdFieldPosition12: db 0
976 <1> MaxPixelClock12: dd 0
977 <1>
978 <1> dw 0140h ; 320x200x32
979 <1> ModeAttributes13: dw MODE_ATTRIBUTES
980 <1> WinAAttributes13: db WINA_ATTRIBUTES
981 <1> WinBAttributes13: db 0
982 <1> WinGranularity13: dw VBE_DISPI_BANK_SIZE_KB
983 <1> WinSize13: dw VBE_DISPI_BANK_SIZE_KB
984 <1> WinASegment13: dw VGAMEM_GRAPH
985 <1> WinBSegment13: dw 0000h
986 <1> WinFuncPtr13: dd 0
987 <1> BytesPerScanLine13: dw 1280
988 <1> XResolution13: dw 320
989 <1> YResolution13: dw 200
990 <1> XCharSize13: db 8
991 <1> YCharSize13: db 16
992 <1> NumberOfPlanes13: db 1
993 <1> BitsPerPixel13: db 32
994 <1> NumberOfBanks13: db 4
995 <1> MemoryModel13: db VBE_MEMORYMODEL_DIRECT_COLOR
996 <1> BankSize13: db 0
997 <1> NumberOfImagePages13: db 64
998 <1> Reserved_page13: db 0
999 <1> RedMaskSize13: db 8
1000 <1> RedFieldPosition13: db 16
1001 <1> GreenMaskSize13: db 8
1002 <1> GreenFieldPosition13: db 8
1003 <1> BlueMaskSize13: db 8
1004 <1> BlueFieldPosition13: db 0
1005 <1> RsvdMaskSize13: db 8
1006 <1> RsvdFieldPosition13: db 24
1007 <1> DirectColorModeInfo13: db VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
1008 <1> PhysBasePtr13: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
1009 <1> OffScreenMemOffset13: dd 0
1010 <1> OffScreenMemSize13: dw 0
1011 <1> LinBytesPerScanLine13: dw 1280
1012 <1> BnkNumberOfPages13: db 0
1013 <1> LinNumberOfPages13: db 0
1014 <1> LinRedMaskSize13: db 8
1015 <1> LinRedFieldPosition13: db 16
1016 <1> LinGreenMaskSize13: db 8
1017 <1> LinGreenFieldPosition13: db 8
1018 <1> LinBlueMaskSize13: db 8
1019 <1> LinBlueFieldPosition13: db 0
1020 <1> LinRsvdMaskSize13: db 8
1021 <1> LinRsvdFieldPosition13: db 24
1022 <1> MaxPixelClock13: dd 0
1023 <1>
1024 <1> dw 0141h ; 640x400x32
1025 <1> ModeAttributes14: dw MODE_ATTRIBUTES
1026 <1> WinAAttributes14: db WINA_ATTRIBUTES
1027 <1> WinBAttributes14: db 0
1028 <1> WinGranularity14: dw VBE_DISPI_BANK_SIZE_KB
1029 <1> WinSize14: dw VBE_DISPI_BANK_SIZE_KB
1030 <1> WinASegment14: dw VGAMEM_GRAPH
1031 <1> WinBSegment14: dw 0000h
1032 <1> WinFuncPtr14: dd 0
1033 <1> BytesPerScanLine14: dw 2560
1034 <1> XResolution14: dw 640
1035 <1> YResolution14: dw 400
1036 <1> XCharSize14: db 8
1037 <1> YCharSize14: db 16
1038 <1> NumberOfPlanes14: db 1
1039 <1> BitsPerPixel14: db 32
1040 <1> NumberOfBanks14: db 16
1041 <1> MemoryModel14: db VBE_MEMORYMODEL_DIRECT_COLOR
1042 <1> BankSize14: db 0
1043 <1> NumberOfImagePages14: db 15
1044 <1> Reserved_page14: db 0
1045 <1> RedMaskSize14: db 8
1046 <1> RedFieldPosition14: db 16
1047 <1> GreenMaskSize14: db 8
1048 <1> GreenFieldPosition14: db 8
1049 <1> BlueMaskSize14: db 8
1050 <1> BlueFieldPosition14: db 0
1051 <1> RsvdMaskSize14: db 8
1052 <1> RsvdFieldPosition14: db 24
1053 <1> DirectColorModeInfo14: db VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
1054 <1> PhysBasePtr14: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
1055 <1> OffScreenMemOffset14: dd 0
1056 <1> OffScreenMemSize14: dw 0
1057 <1> LinBytesPerScanLine14: dw 2560
1058 <1> BnkNumberOfPages14: db 0
1059 <1> LinNumberOfPages14: db 0
1060 <1> LinRedMaskSize14: db 8
1061 <1> LinRedFieldPosition14: db 16
1062 <1> LinGreenMaskSize14: db 8
1063 <1> LinGreenFieldPosition14: db 8
1064 <1> LinBlueMaskSize14: db 8
1065 <1> LinBlueFieldPosition14: db 0
1066 <1> LinRsvdMaskSize14: db 8
1067 <1> LinRsvdFieldPosition14: db 24
1068 <1> MaxPixelClock14: dd 0
1069 <1>
1070 <1> dw 0142 ; 640x480x32
1071 <1> ModeAttributes15: dw MODE_ATTRIBUTES
1072 <1> WinAAttributes15: db WINA_ATTRIBUTES
1073 <1> WinBAttributes15: db 0
1074 <1> WinGranularity15: dw VBE_DISPI_BANK_SIZE_KB
1075 <1> WinSize15: dw VBE_DISPI_BANK_SIZE_KB
1076 <1> WinASegment15: dw VGAMEM_GRAPH
1077 <1> WinBSegment15: dw 0000h
1078 <1> WinFuncPtr15: dd 0
1079 <1> BytesPerScanLine15: dw 2560
1080 <1> XResolution15: dw 640
1081 <1> YResolution15: dw 480
1082 <1> XCharSize15: db 8
1083 <1> YCharSize15: db 16
1084 <1> NumberOfPlanes15: db 1
1085 <1> BitsPerPixel15: db 32
1086 <1> NumberOfBanks15: db 19
1087 <1> MemoryModel15: db VBE_MEMORYMODEL_DIRECT_COLOR
1088 <1> BankSize15: db 0
1089 <1> NumberOfImagePages15: db 12
1090 <1> Reserved_page15: db 0
1091 <1> RedMaskSize15: db 8
1092 <1> RedFieldPosition15: db 16
1093 <1> GreenMaskSize15: db 8

```

```

1094 <1> GreenFieldPosition15: db 8
1095 <1> BlueMaskSize15: db 8
1096 <1> BlueFieldPosition15: db 0
1097 <1> RsvdMaskSize15: db 8
1098 <1> RsvdFieldPosition15: db 24
1099 <1> DirectColorModeInfo15: db VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE,
1100 <1> PhysBasePtr15: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS,
1101 <1> OffScreenMemOffset15: dd 0
1102 <1> OffScreenMemSize15: dw 0
1103 <1> LinBytesPerScanLine15: dw 2560
1104 <1> BnkNumberOfPages15: db 0
1105 <1> LinNumberOfPages15: db 0
1106 <1> LinRedMaskSize15: db 8
1107 <1> LinRedFieldPosition15: db 16
1108 <1> LinGreenMaskSize15: db 8
1109 <1> LinGreenFieldPosition15: db 8
1110 <1> LinBlueMaskSize15: db 8
1111 <1> LinBlueFieldPosition15: db 0
1112 <1> LinRsvdMaskSize15: db 8
1113 <1> LinRsvdFieldPosition15: db 24
1114 <1> MaxPixelClock15: dd 0
1115 <1>
1116 <1> dw 0143h ; 800x600x32
1117 <1> ModeAttributes16: dw MODE_ATTRIBUTES
1118 <1> WinAAttributes16: db WINA_ATTRIBUTES
1119 <1> WinBAttributes16: db 0
1120 <1> WinGranularity16: dw VBE_DISPI_BANK_SIZE_KB
1121 <1> WinSize16: dw VBE_DISPI_BANK_SIZE_KB
1122 <1> WinASegment16: dw VGAMEM_GRAPH
1123 <1> WinBSegment16: dw 0000h
1124 <1> WinFuncPtr16: dd 0
1125 <1> BytesPerScanLine16: dw 3200
1126 <1> XResolution16: dw 800
1127 <1> YResolution16: dw 600
1128 <1> XCharSize16: db 8
1129 <1> YCharSize16: db 16
1130 <1> NumberOfPlanes16: db 1
1131 <1> BitsPerPixel16: db 32
1132 <1> NumberOfBanks16: db 30
1133 <1> MemoryModel16: db VBE_MEMORYMODEL_DIRECT_COLOR
1134 <1> BankSize16: db 0
1135 <1> NumberOfImagePages16: db 7
1136 <1> Reserved_page16: db 0
1137 <1> RedMaskSize16: db 8
1138 <1> RedFieldPosition16: db 16
1139 <1> GreenMaskSize16: db 8
1140 <1> GreenFieldPosition16: db 8
1141 <1> BlueMaskSize16: db 8
1142 <1> BlueFieldPosition16: db 0
1143 <1> RsvdMaskSize16: db 8
1144 <1> RsvdFieldPosition16: db 24
1145 <1> DirectColorModeInfo16: db VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE,
1146 <1> PhysBasePtr16: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS,
1147 <1> OffScreenMemOffset16: dd 0
1148 <1> OffScreenMemSize16: dw 0
1149 <1> LinBytesPerScanLine16: dw 3200
1150 <1> BnkNumberOfPages16: db 0
1151 <1> LinNumberOfPages16: db 0
1152 <1> LinRedMaskSize16: db 8
1153 <1> LinRedFieldPosition16: db 16
1154 <1> LinGreenMaskSize16: db 8
1155 <1> LinGreenFieldPosition16: db 8
1156 <1> LinBlueMaskSize16: db 8
1157 <1> LinBlueFieldPosition16: db 0
1158 <1> LinRsvdMaskSize16: db 8
1159 <1> LinRsvdFieldPosition16: db 24
1160 <1> MaxPixelClock16: dd 0
1161 <1>
1162 <1> dw 0144h ; 1024x768x32
1163 <1> ModeAttributes17: dw MODE_ATTRIBUTES
1164 <1> WinAAttributes17: db WINA_ATTRIBUTES
1165 <1> WinBAttributes17: db 0
1166 <1> WinGranularity17: dw VBE_DISPI_BANK_SIZE_KB
1167 <1> WinSize17: dw VBE_DISPI_BANK_SIZE_KB
1168 <1> WinASegment17: dw VGAMEM_GRAPH
1169 <1> WinBSegment17: dw 0000h
1170 <1> WinFuncPtr17: dd 0
1171 <1> BytesPerScanLine17: dw 4096
1172 <1> XResolution17: dw 1024
1173 <1> YResolution17: dw 768
1174 <1> XCharSize17: db 8
1175 <1> YCharSize17: db 16
1176 <1> NumberOfPlanes17: db 1
1177 <1> BitsPerPixel17: db 32
1178 <1> NumberOfBanks17: db 48
1179 <1> MemoryModel17: db VBE_MEMORYMODEL_DIRECT_COLOR
1180 <1> BankSize17: db 0
1181 <1> NumberOfImagePages17: db 4
1182 <1> Reserved_page17: db 0
1183 <1> RedMaskSize17: db 8
1184 <1> RedFieldPosition17: db 16
1185 <1> GreenMaskSize17: db 8
1186 <1> GreenFieldPosition17: db 8
1187 <1> BlueMaskSize17: db 8
1188 <1> BlueFieldPosition17: db 0
1189 <1> RsvdMaskSize17: db 8
1190 <1> RsvdFieldPosition17: db 24
1191 <1> DirectColorModeInfo17: db VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE,
1192 <1> PhysBasePtr17: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS,
1193 <1> OffScreenMemOffset17: dd 0
1194 <1> OffScreenMemSize17: dw 0
1195 <1> LinBytesPerScanLine17: dw 4096
1196 <1> BnkNumberOfPages17: db 0
1197 <1> LinNumberOfPages17: db 0
1198 <1> LinRedMaskSize17: db 8
1199 <1> LinRedFieldPosition17: db 16
1200 <1> LinGreenMaskSize17: db 8
1201 <1> LinGreenFieldPosition17: db 8
1202 <1> LinBlueMaskSize17: db 8
1203 <1> LinBlueFieldPosition17: db 0
1204 <1> LinRsvdMaskSize17: db 8
1205 <1> LinRsvdFieldPosition17: db 24
1206 <1> MaxPixelClock17: dd 0
1207 <1>
1208 <1> dw 0146h ; 320x200x8
1209 <1> ModeAttributes18: dw MODE_ATTRIBUTES
1210 <1> WinAAttributes18: db WINA_ATTRIBUTES
1211 <1> WinBAttributes18: db 0
1212 <1> WinGranularity18: dw VBE_DISPI_BANK_SIZE_KB
1213 <1> WinSize18: dw VBE_DISPI_BANK_SIZE_KB
1214 <1> WinASegment18: dw VGAMEM_GRAPH
1215 <1> WinBSegment18: dw 0000h
1216 <1> WinFuncPtr18: dd 0
1217 <1> BytesPerScanLine18: dw 320

```

```

1218 <1> XResolution18: dw 320
1219 <1> YResolution18: dw 200
1220 <1> XCharSize18: db 8
1221 <1> YCharSize18: db 16
1222 <1> NumberOfPlanes18: db 1
1223 <1> BitsPerPixel18: db 8
1224 <1> NumberOfBanks18: db 1
1225 <1> MemoryModel18: db VBE_MEMORYMODEL_PACKED_PIXEL
1226 <1> BankSize18: db 0
1227 <1> NumberOfImagePages18: db 255 ; 261 in vbetables.h (03/01/2020) !
1228 <1> Reserved_page18: db 0
1229 <1> RedMaskSize18: db 0
1230 <1> RedFieldPosition18: db 0
1231 <1> GreenMaskSize18: db 0
1232 <1> GreenFieldPosition18: db 0
1233 <1> BlueMaskSize18: db 0
1234 <1> BlueFieldPosition18: db 0
1235 <1> RsvdMaskSize18: db 0
1236 <1> RsvdFieldPosition18: db 0
1237 <1> DirectColorModeInfo18: db 0
1238 <1> PhysBasePtr18: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
1239 <1> OffScreenMemOffset18: dd 0
1240 <1> OffScreenMemSize18: dw 0
1241 <1> LinBytesPerScanLine18: dw 320
1242 <1> BnkNumberOfPages18: db 0
1243 <1> LinNumberOfPages18: db 0
1244 <1> LinRedMaskSize18: db 0
1245 <1> LinRedFieldPosition18: db 0
1246 <1> LinGreenMaskSize18: db 0
1247 <1> LinGreenFieldPosition18: db 0
1248 <1> LinBlueMaskSize18: db 0
1249 <1> LinBlueFieldPosition18: db 0
1250 <1> LinRsvdMaskSize18: db 0
1251 <1> LinRsvdFieldPosition18: db 0
1252 <1> MaxPixelClock18: dd 0
1253 <1>
1254 <1>
1255 <1> ModeAttributes19: dw 018Dh ; 1280x720x16
1256 <1> WinAAttributes19: db MODE_ATTRIBUTES
1257 <1> WinBAttributes19: db WINA_ATTRIBUTES
1258 <1> WinGranularity19: dw VBE_DISPI_BANK_SIZE_KB
1259 <1> WinSize19: dw VBE_DISPI_BANK_SIZE_KB
1260 <1> WinASegment19: dw VGAMEM_GRAPH
1261 <1> WinBSegment19: dw 0000h
1262 <1> WinFuncPtr19: dd 0
1263 <1> BytesPerScanLine19: dw 2560
1264 <1> XResolution19: dw 1280
1265 <1> YResolution19: dw 720
1266 <1> XCharSize19: db 8
1267 <1> YCharSize19: db 16
1268 <1> NumberOfPlanes19: db 1
1269 <1> BitsPerPixel19: db 16
1270 <1> NumberOfBanks19: db 29
1271 <1> MemoryModel19: db VBE_MEMORYMODEL_DIRECT_COLOR
1272 <1> BankSize19: db 0
1273 <1> NumberOfImagePages19: db 8
1274 <1> Reserved_page19: db 0
1275 <1> RedMaskSize19: db 5
1276 <1> RedFieldPosition19: db 11
1277 <1> GreenMaskSize19: db 6
1278 <1> GreenFieldPosition19: db 5
1279 <1> BlueMaskSize19: db 5
1280 <1> BlueFieldPosition19: db 0
1281 <1> RsvdMaskSize19: db 0
1282 <1> RsvdFieldPosition19: db 0
1283 <1> DirectColorModeInfo19: db 0
1284 <1> PhysBasePtr19: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
1285 <1> OffScreenMemOffset19: dd 0
1286 <1> OffScreenMemSize19: dw 0
1287 <1> LinBytesPerScanLine19: dw 2560
1288 <1> BnkNumberOfPages19: db 0
1289 <1> LinNumberOfPages19: db 0
1290 <1> LinRedMaskSize19: db 5
1291 <1> LinRedFieldPosition19: db 11
1292 <1> LinGreenMaskSize19: db 6
1293 <1> LinGreenFieldPosition19: db 5
1294 <1> LinBlueMaskSize19: db 5
1295 <1> LinBlueFieldPosition19: db 0
1296 <1> LinRsvdMaskSize19: db 0
1297 <1> LinRsvdFieldPosition19: db 0
1298 <1> MaxPixelClock19: dd 0
1299 <1>
1300 <1>
1301 <1> ModeAttributes20: dw 018Eh ; 1280x720x24
1302 <1> WinAAttributes20: db MODE_ATTRIBUTES
1303 <1> WinBAttributes20: db WINA_ATTRIBUTES
1304 <1> WinGranularity20: dw VBE_DISPI_BANK_SIZE_KB
1305 <1> WinSize20: dw VBE_DISPI_BANK_SIZE_KB
1306 <1> WinASegment20: dw VGAMEM_GRAPH
1307 <1> WinBSegment20: dw 0000h
1308 <1> WinFuncPtr20: dd 0
1309 <1> BytesPerScanLine20: dw 3840
1310 <1> XResolution20: dw 1280
1311 <1> YResolution20: dw 720
1312 <1> XCharSize20: db 8
1313 <1> YCharSize20: db 16
1314 <1> NumberOfPlanes20: db 1
1315 <1> BitsPerPixel20: db 24
1316 <1> NumberOfBanks20: db 43
1317 <1> MemoryModel20: db VBE_MEMORYMODEL_DIRECT_COLOR
1318 <1> BankSize20: db 0
1319 <1> NumberOfImagePages20: db 5
1320 <1> Reserved_page20: db 0
1321 <1> RedMaskSize20: db 8
1322 <1> RedFieldPosition20: db 16
1323 <1> GreenMaskSize20: db 8
1324 <1> GreenFieldPosition20: db 8
1325 <1> BlueMaskSize20: db 8
1326 <1> BlueFieldPosition20: db 0
1327 <1> RsvdMaskSize20: db 0
1328 <1> RsvdFieldPosition20: db 0
1329 <1> DirectColorModeInfo20: db 0
1330 <1> PhysBasePtr20: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
1331 <1> OffScreenMemOffset20: dd 0
1332 <1> OffScreenMemSize20: dw 0
1333 <1> LinBytesPerScanLine20: dw 3840
1334 <1> BnkNumberOfPages20: db 0
1335 <1> LinNumberOfPages20: db 0
1336 <1> LinRedMaskSize20: db 8
1337 <1> LinRedFieldPosition20: db 16
1338 <1> LinGreenMaskSize20: db 8
1339 <1> LinGreenFieldPosition20: db 8
1340 <1> LinBlueMaskSize20: db 8
1341 <1> LinBlueFieldPosition20: db 0

```



```

1342 <1> LinRsvdMaskSize20: db 0
1343 <1> LinRsvdFieldPosition20: db 0
1344 <1> MaxPixelClock20: dd 0
1345 <1>
1346 <1> dw 018Fh ; 1280x720x32
1347 <1> ModeAttributes21: dw MODE_ATTRIBUTES
1348 <1> WinAAttributes21: db WINA_ATTRIBUTES
1349 <1> WinBAttributes21: db 0
1350 <1> WinGranularity21: dw VBE_DISPI_BANK_SIZE_KB
1351 <1> WinSize21: dw VBE_DISPI_BANK_SIZE_KB
1352 <1> WinASegment21: dw VGAMEM_GRAPH
1353 <1> WinBSegment21: dw 0000h
1354 <1> WinFuncPtr21: dd 0
1355 <1> BytesPerScanLine21: dw 5120
1356 <1> XResolution21: dw 1280
1357 <1> YResolution21: dw 720
1358 <1> XCharSize21: db 8
1359 <1> YCharSize21: db 16
1360 <1> NumberOfPlanes21: db 1
1361 <1> BitsPerPixel21: db 32
1362 <1> NumberOfBanks21: db 57
1363 <1> MemoryModel21: db VBE_MEMORYMODEL_DIRECT_COLOR
1364 <1> BankSize21: db 0
1365 <1> NumberOfImagePages21: db 3
1366 <1> Reserved_page21: db 0
1367 <1> RedMaskSize21: db 8
1368 <1> RedFieldPosition21: db 16
1369 <1> GreenMaskSize21: db 8
1370 <1> GreenFieldPosition21: db 8
1371 <1> BlueMaskSize21: db 8
1372 <1> BlueFieldPosition21: db 0
1373 <1> RsvdMaskSize21: db 8
1374 <1> RsvdFieldPosition21: db 24
1375 <1> DirectColorModeInfo21: db VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
1376 <1> PhysBasePtr21: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
1377 <1> OffScreenMemOffset21: dd 0
1378 <1> OffScreenMemSize21: dw 0
1379 <1> LinBytesPerScanLine21: dw 5120
1380 <1> BnkNumberOfPages21: db 0
1381 <1> LinNumberOfPages21: db 0
1382 <1> LinRedMaskSize21: db 8
1383 <1> LinRedFieldPosition21: db 16
1384 <1> LinGreenMaskSize21: db 8
1385 <1> LinGreenFieldPosition21: db 8
1386 <1> LinBlueMaskSize21: db 8
1387 <1> LinBlueFieldPosition21: db 0
1388 <1> LinRsvdMaskSize21: db 8
1389 <1> LinRsvdFieldPosition21: db 24
1390 <1> MaxPixelClock21: dd 0
1391 <1>
1392 <1> dw 0190h ; 1920x1080x16
1393 <1> ModeAttributes22: dw MODE_ATTRIBUTES
1394 <1> WinAAttributes22: db WINA_ATTRIBUTES
1395 <1> WinBAttributes22: db 0
1396 <1> WinGranularity22: dw VBE_DISPI_BANK_SIZE_KB
1397 <1> WinSize22: dw VBE_DISPI_BANK_SIZE_KB
1398 <1> WinASegment22: dw VGAMEM_GRAPH
1399 <1> WinBSegment22: dw 0000h
1400 <1> WinFuncPtr22: dd 0
1401 <1> BytesPerScanLine22: dw 3840
1402 <1> XResolution22: dw 1920
1403 <1> YResolution22: dw 1080
1404 <1> XCharSize22: db 8
1405 <1> YCharSize22: db 16
1406 <1> NumberOfPlanes22: db 1
1407 <1> BitsPerPixel22: db 16
1408 <1> NumberOfBanks22: db 64
1409 <1> MemoryModel22: db VBE_MEMORYMODEL_DIRECT_COLOR,
1410 <1> BankSize22: db 0
1411 <1> NumberOfImagePages22: db 3
1412 <1> Reserved_page22: db 0
1413 <1> RedMaskSize22: db 5
1414 <1> RedFieldPosition22: db 11
1415 <1> GreenMaskSize22: db 6
1416 <1> GreenFieldPosition22: db 5
1417 <1> BlueMaskSize22: db 5
1418 <1> BlueFieldPosition22: db 0
1419 <1> RsvdMaskSize22: db 0
1420 <1> RsvdFieldPosition22: db 0
1421 <1> DirectColorModeInfo22: db 0
1422 <1> PhysBasePtr22: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
1423 <1> OffScreenMemOffset22: dd 0
1424 <1> OffScreenMemSize22: dw 0
1425 <1> LinBytesPerScanLine22: dw 3840
1426 <1> BnkNumberOfPages22: db 0
1427 <1> LinNumberOfPages22: db 0
1428 <1> LinRedMaskSize22: db 5
1429 <1> LinRedFieldPosition22: db 11
1430 <1> LinGreenMaskSize22: db 6
1431 <1> LinGreenFieldPosition22: db 5
1432 <1> LinBlueMaskSize22: db 5
1433 <1> LinBlueFieldPosition22: db 0
1434 <1> LinRsvdMaskSize22: db 0
1435 <1> LinRsvdFieldPosition22: db 0
1436 <1> MaxPixelClock22: dd 0
1437 <1>
1438 <1> dw 0191h ; 1920x1080x24
1439 <1> ModeAttributes23: dw MODE_ATTRIBUTES
1440 <1> WinAAttributes23: db WINA_ATTRIBUTES
1441 <1> WinBAttributes23: db 0
1442 <1> WinGranularity23: dw VBE_DISPI_BANK_SIZE_KB
1443 <1> WinSize23: dw VBE_DISPI_BANK_SIZE_KB
1444 <1> WinASegment23: dw VGAMEM_GRAPH
1445 <1> WinBSegment23: dw 0000h
1446 <1> WinFuncPtr23: dd 0
1447 <1> BytesPerScanLine23: dw 5760
1448 <1> XResolution23: dw 1920
1449 <1> YResolution23: dw 1080
1450 <1> XCharSize23: db 8
1451 <1> YCharSize23: db 16
1452 <1> NumberOfPlanes23: db 1
1453 <1> BitsPerPixel23: db 24
1454 <1> NumberOfBanks23: db 95
1455 <1> MemoryModel23: db VBE_MEMORYMODEL_DIRECT_COLOR
1456 <1> BankSize23: db 0
1457 <1> NumberOfImagePages23: db 1
1458 <1> Reserved_page23: db 0
1459 <1> RedMaskSize23: db 8
1460 <1> RedFieldPosition23: db 16
1461 <1> GreenMaskSize23: db 8
1462 <1> GreenFieldPosition23: db 8
1463 <1> BlueMaskSize23: db 8
1464 <1> BlueFieldPosition23: db 0
1465 <1> RsvdMaskSize23: db 0

```

```

1466 <1> RsvdFieldPosition23: db 0
1467 <1> DirectColorModeInfo23: db 0
1468 <1> PhysBasePtr23: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
1469 <1> OffScreenMemOffset23: dd 0
1470 <1> OffScreenMemSize23: dw 0
1471 <1> LinBytesPerScanLine23: dw 5760
1472 <1> BnkNumberOfPages23: db 0
1473 <1> LinNumberOfPages23: db 0
1474 <1> LinRedMaskSize23: db 8
1475 <1> LinRedFieldPosition23: db 16
1476 <1> LinGreenMaskSize23: db 8
1477 <1> LinGreenFieldPosition23: db 8
1478 <1> LinBlueMaskSize23: db 8
1479 <1> LinBlueFieldPosition23: db 0
1480 <1> LinRsvdMaskSize23: db 0
1481 <1> LinRsvdFieldPosition23: db 0
1482 <1> MaxPixelClock23: dd 0
1483 <1>
1484 <1> dw 0192h ; 1920x1080x32
1485 <1> ModeAttributes24: dw MODE_ATTRIBUTES
1486 <1> WinAAttributes24: db WINA_ATTRIBUTES
1487 <1> WinBAttributes24: db 0
1488 <1> WinGranularity24: dw VBE_DISPI_BANK_SIZE_KB
1489 <1> WinSize24: dw VBE_DISPI_BANK_SIZE_KB
1490 <1> WinASegment24: dw VGAMEM_GRAPH
1491 <1> WinBSegment24: dw 0000h
1492 <1> WinFuncPtr24: dd 0
1493 <1> BytesPerScanLine24: dw 7680
1494 <1> XResolution24: dw 1920
1495 <1> YResolution24: dw 1080
1496 <1> XCharSize24: db 8
1497 <1> YCharSize24: db 16
1498 <1> NumberOfPlanes24: db 1
1499 <1> BitsPerPixel24: db 32
1500 <1> NumberOfBanks24: db 127
1501 <1> MemoryModel24: db VBE_MEMORYMODEL_DIRECT_COLOR
1502 <1> BanksSize24: db 0
1503 <1> NumberOfImagePages24: db 1
1504 <1> Reserved_page24: db 0
1505 <1> RedMaskSize24: db 8
1506 <1> RedFieldPosition24: db 16
1507 <1> GreenMaskSize24: db 8
1508 <1> GreenFieldPosition24: db 8
1509 <1> BlueMaskSize24: db 8
1510 <1> BlueFieldPosition24: db 0
1511 <1> RsvdMaskSize24: db 8
1512 <1> RsvdFieldPosition24: db 24
1513 <1> DirectColorModeInfo24: db VBE_DIRECTCOLOR_RESERVED_BITS_AVAILABLE
1514 <1> PhysBasePtr24: dd VBE_DISPI_LFB_PHYSICAL_ADDRESS
1515 <1> OffScreenMemOffset24: dd 0
1516 <1> OffScreenMemSize24: dw 0
1517 <1> LinBytesPerScanLine24: dw 7680
1518 <1> BnkNumberOfPages24: db 0
1519 <1> LinNumberOfPages24: db 0
1520 <1> LinRedMaskSize24: db 8
1521 <1> LinRedFieldPosition24: db 16
1522 <1> LinGreenMaskSize24: db 8
1523 <1> LinGreenFieldPosition24: db 8
1524 <1> LinBlueMaskSize24: db 8
1525 <1> LinBlueFieldPosition24: db 0
1526 <1> LinRsvdMaskSize24: db 8
1527 <1> LinRsvdFieldPosition24: db 24
1528 <1> MaxPixelClock24: dd 0
1529 <1>
1530 <1> VBE_VESA_MODE_END_OF_LIST: dw 0
1531 <1>
1532 <1> %endif
1533 <1>
1534 <1> ; 24/11/2020
1535 <1>
1536 <1> direct_color_fields:
1537 <1> ; 24/11/2020
1538 <1>
1539 <1> ; (vbetables-gen.c)
1540 <1> ; // Direct Color fields
1541 <1> ; (required for direct/6 and YUV/7 memory models)
1542 <1> ; switch(pm->depth) {
1543 <1>
1544 <1> ;case 8:
1545 00006C6A 00 <1> r_size_8: db 0
1546 00006C6B 00 <1> r_pos_8: db 0
1547 00006C6C 00 <1> g_size_8: db 0
1548 00006C6D 00 <1> g_pos_8: db 0
1549 00006C6E 00 <1> b_size_8: db 0
1550 00006C6F 00 <1> b_pos_8: db 0
1551 00006C70 00 <1> a_size_8: db 0
1552 00006C71 00 <1> a_pos_8: db 0
1553 <1>
1554 <1> ;case 16:
1555 00006C72 05 <1> r_size_16: db 5
1556 00006C73 08 <1> r_pos_16: db 11
1557 00006C74 06 <1> g_size_16: db 6
1558 00006C75 05 <1> g_pos_16: db 5
1559 00006C76 05 <1> b_size_16: db 5
1560 00006C77 00 <1> b_pos_16: db 0
1561 00006C78 00 <1> a_size_16: db 0
1562 00006C79 00 <1> a_pos_16: db 0
1563 <1>
1564 <1> ;case 24:
1565 00006C7A 08 <1> r_size_24: db 8
1566 00006C7B 10 <1> r_pos_24: db 16
1567 00006C7C 08 <1> g_size_24: db 8
1568 00006C7D 08 <1> g_pos_24: db 8
1569 00006C7E 08 <1> b_size_24: db 8
1570 00006C7F 00 <1> b_pos_24: db 0
1571 00006C80 00 <1> a_size_24: db 0
1572 00006C81 00 <1> a_pos_24: db 0
1573 <1>
1574 <1> ;case 32:
1575 00006C82 08 <1> r_size_32: db 8
1576 00006C83 10 <1> r_pos_32: db 16
1577 00006C84 08 <1> g_size_32: db 8
1578 00006C85 08 <1> g_pos_32: db 8
1579 00006C86 08 <1> b_size_32: db 8
1580 00006C87 00 <1> b_pos_32: db 0
1581 00006C88 08 <1> a_size_32: db 8
1582 00006C89 18 <1> a_pos_32: db 24
3425 <1> ;%include 'diskdata.s' ; DISK (BIOS) DATA (initialized)
3426 <1> ;;;
3427 <1>
3428 <1> Align 2
3429 <1>
3430 <1> %include 'sysdefs.s' ; 24/01/2015
3431 <1> ; *****

```

```

2      <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.5) - SYSTEM DEFINITIONS : sysdefs.s
3      <1> ; -----
4      <1> ; Last Update: 23/07/2022 (Previous: 31/12/2017)
5      <1> ; -----
6      <1> ; Beginning: 24/01/2016
7      <1> ; -----
8      <1> ; Assembler: NASM version 2.11 (trdos386.s)
9      <1> ; -----
10     <1> ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
11     <1> ; sysdefs.inc (14/11/2015)
12     <1> ; *****
13     <1> ;
14     <1> ; Retro UNIX 386 v1 Kernel - SYSDEFS.INC
15     <1> ; Last Modification: 14/11/2015
16     <1> ;
17     <1> ; //////////////////////////////////////////////////
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> nttyequ    8 ; 8+1 -> 8 (10/05/2013)
34     <1> nbufilequ  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> ;sdsegmnt equ 6C0h ; 256*16 bytes (swap data segment size for 16 processes)
56     <1> ; 19/04/2013 Retro UNIX 8086 v1 feaure only !
57     <1> ;sdsegmnt equ 740h ; swap data segment (for user structures and registers)
58     <1> ;
59     <1> ; 30/08/2013
60     <1> time_count equ 4 ; 10 --> 4 01/02/2014
61     <1> ;
62     <1> ; 05/02/2014
63     <1> ; process status
64     <1> ;SFRREE equ 0
65     <1> ;SRUN equ 1
66     <1> ;SWAIT equ 2
67     <1> ;SZOMB equ 3
68     <1> ;SSLEEP equ 4 ; Retro UNIX 8086 v1 extension (for sleep and wakeup)
69     <1> ;
70     <1> ; 09/03/2015
71     <1> userdata equ 80000h ; user structure data address for current user ; temporary
72     <1> swap_queue equ 90000h - 2000h ; swap queue address ; temporary
73     <1> swap_alloc_table equ 0D0000h ; swap allocation table address ; temporary
74     <1> ;
75     <1> ; 17/09/2015
76     <1> ESPACE equ 48 ; [u.usp] (at 'sysent') - [u.sp] value for error return
77     <1> ;
78     <1> ; 31/12/2017
79     <1> ; 19/02/2017
80     <1> ; 15/10/2016
81     <1> ; 20/05/2016
82     <1> ; 19/05/2016
83     <1> ; 18/05/2016
84     <1> ; 29/04/2016
85     <1> ; TRDOS 386 (TRDOS v2.0) system calls - temporary List
86     <1> ; 14/07/2013 - 21/09/2015 (Retro UNIX 8086 & 386 system calls)
87     <1> _ver equ 0 ; Get TRDOS version (v2.0)
88     <1> _exit equ 1
89     <1> _fork equ 2
90     <1> _read equ 3
91     <1> _write equ 4
92     <1> _open equ 5
93     <1> _close equ 6
94     <1> _wait equ 7
95     <1> _creat equ 8
96     <1> _rename equ 9 ; TRDOS 386, Rename File (31/12/2017)
97     <1> _delete equ 10 ; TRDOS 386, Delete File (29/12/2017)
98     <1> _exec equ 11
99     <1> _chdir equ 12
100    <1> _time equ 13 ; TRDOS 386, Get Sys Date&Time (30/12/2017)
101    <1> _mkdir equ 14
102    <1> _chmod equ 15 ; TRDOS 386, Change Attributes (30/12/2017)
103    <1> _rmdir equ 16 ; TRDOS 386, Remove Directory (29/12/2017)
104    <1> _break equ 17
105    <1> _drive equ 18 ; TRDOS 386, Get/Set Current Drv (30/12/2017)
106    <1> _seek equ 19
107    <1> _tell equ 20
108    <1> _memequ 21 ; TRDOS 386, Get Total&Free Mem (31/12/2017)
109    <1> _prompt equ 22 ; TRDOS 386, Change Cmd Prompt (31/12/2017)
110    <1> _path equ 23 ; TRDOS 386, Get/Set Run Path (31/12/2017)
111    <1> _envequ 24 ; TRDOS 386, Get/Set Env Vars (31/12/2017)
112    <1> _stime equ 25 ; TRDOS 386, Set Sys Date&Time (30/12/2017)
113    <1> _quit equ 26
114    <1> _intr equ 27
115    <1> _direqu 28 ; TRDOS 386, Get Curr Drive&Dir (30/12/2017)
116    <1> _emt equ 29
117    <1> _ldrvt equ 30 ; TRDOS 386, Get Logical DOS DDT (30/12/2017)
118    <1> _video equ 31 ; TRDOS 386 Video Functions (16/05/2016)
119    <1> _audio equ 32 ; TRDOS 386 Video Functions (16/05/2016)
120    <1> _timer equ 33 ; TRDOS 386 Timer Functions (18/05/2016)
121    <1> _sleep equ 34 ; Retro UNIX 8086 v1 feature only !
122    <1> _msgequ 35 ; Retro UNIX 386 v1 feature only !
123    <1> _geterr equ 36 ; Retro UNIX 386 v1 feature only !
124    <1> _fpsave equ 37 ; TRDOS 386 FPU state option (28/02/2017)
125    <1> _pri equ 38 ; change priority - TRDOS 386 (20/05/2016)

```

```

126 <1> _rele equ 39 ; TRDOS 386 (19/05/2016)
127 <1> _fffequ 40 ; Find First File - TRDOS 386 (15/10/2016)
128 <1> _fnfequ 41 ; Find Next File - TRDOS 386 (15/10/2016)
129 <1> _alloc equ 42 ; Allocate memory - TRDOS 386 (19/02/2017)
130 <1> ; TRDOS 386 (19/02/2017) DMA buff fuctions
131 <1> _dalloc equ 43 ; Deallocate mem - TRDOS 386 (19/02/2017)
132 <1> _calbac equ 44 ; Set IRQ callback - TRDOS 386 (20/02/2017)
133 <1> _dmaequ 45 ; DMA service - TRDOS 386 (20/08/2017)
134 <1>
135 <1> %macro sys 1-4
136 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
137 <1> ; 03/09/2015
138 <1> ; 13/04/2015
139 <1> ; Retro UNIX 386 v1 system call.
140 <1> %if %0 >= 2
141 <1> mov ebx, %2
142 <1> %if %0 >= 3
143 <1> mov ecx, %3
144 <1> %if %0 = 4
145 <1> mov edx, %4
146 <1> %endif
147 <1> %endif
148 <1> %endif
149 <1> mov eax, %1
150 <1> ;int 30h
151 <1> int 40h ; TRDOS 386 (TRDOS v2.0)
152 <1> %endmacro
153 <1>
154 <1> ; TRDOS 386 system calls, interrupt number
155 <1> ; 25/12/2016
156 <1> SYSCALL_INT_NUM equ '40' ; '40h'
157 <1>
158 <1> ; 13/05/2015 - ERROR CODES
159 <1> ERR_FILE_NOT_OPEN equ 10 ; 'file not open !' error
160 <1> ERR_FILE_ACCESS equ 11 ; 'permission denied !' error
161 <1> ; 14/05/2015
162 <1> ERR_DIR_ACCESS equ 11 ; 'permission denied !' error
163 <1> ERR_FILE_NOT_FOUND equ 12 ; 'file not found !' error
164 <1> ERR_TOO_MANY_FILES equ 13 ; 'too many open files !' error
165 <1> ERR_DIR_EXISTS equ 14 ; 'directory already exists !' error
166 <1> ; 16/05/2015
167 <1> ERR_DRV_NOT_RDY equ 15 ; 'drive not ready !' error
168 <1> ; 18/05/2015
169 <1> ERR_DEV_NOT_RDY equ 15 ; 'device not ready !' error
170 <1> ERR_DEV_ACCESS equ 11 ; 'permission denied !' error
171 <1> ERR_DEV_NOT_OPEN equ 10 ; 'device not open !' error
172 <1> ; 07/06/2015
173 <1> ERR_FILE_EOF equ 16 ; 'end of file !' error
174 <1> ERR_DEV_VOL_SIZE equ 16 ; 'out of volume !' error
175 <1> ; 09/06/2015
176 <1> ERR_DRV_READ equ 17 ; 'disk read error !'
177 <1> ERR_DRV_WRITE equ 18 ; 'disk write error !'
178 <1> ; 16/06/2015
179 <1> ERR_NOT_DIR equ 19 ; 'not a (valid) directory !' error
180 <1> ERR_FILE_SIZE equ 20 ; 'file size error !'
181 <1> ; 22/06/2015
182 <1> ERR_NOT_SUPERUSER equ 11 ; 'permission denied !' error
183 <1> ERR_NOT_OWNER equ 11 ; 'permission denied !' error
184 <1> ERR_NOT_FILE equ 11 ; 'permission denied !' error
185 <1> ; 23/06/2015
186 <1> ERR_FILE_EXISTS equ 14 ; 'file already exists !' error
187 <1> ERR_DRV_NOT_SAME equ 21 ; 'not same drive !' error
188 <1> ERR_DIR_NOT_FOUND equ 12 ; 'directory not found !' error
189 <1> ERR_NOT_EXECUTABLE equ 22 ; 'not executable file !' error
190 <1> ; 27/06/2015
191 <1> ERR_INV_PARAMETER equ 23 ; 'invalid parameter !' error
192 <1> ERR_INV_DEV_NAME equ 24 ; 'invalid device name !' error
193 <1> ; 29/06/2015
194 <1> ERR_TIME_OUT equ 25 ; 'time out !' error
195 <1> ERR_DEV_NOT_RESP equ 25 ; 'device not responding !' error
196 <1> ; 10/10/2016
197 <1> ERR_INV_FILE_NAME equ 26 ; 'invalid file name !' error
198 <1> ERR_INV_FLAGS equ 23 ; 'invalid flags !' error
199 <1> ; For code compatibility with previous version of TRDOS (2011)
200 <1> ; (Temporary error codes for current TRDOS 386 -2016- version)
201 <1> ERR_NO_MORE_FILES equ 12 ; 'no more files !' error
202 <1> ERR_PATH_NOT_FOUND equ 3 ; 'path not found !' error
203 <1> ; 'dir not found !' ; TRDOS 8086
204 <1> ERR_NOT_FOUND equ 2 ; 'file not found !' ; TRDOS 8086
205 <1> ERR_DISK_SPACE equ 39 ; 'out of volume !' TRDOS 8086
206 <1> ; 'insufficient disk space !' ; 27h
207 <1> ERR_DISK_WRITE equ 30 ; 'disk write protected !' ; 16/10/2016
208 <1> ERR_ACCESS_DENIED equ 5 ; 'access denied !' ; TRDOS 8086
209 <1> ; 28/02/2017
210 <1> ERR_PERM_DENIED equ 11 ; 'permission denied !' error
211 <1> ; 18/05/2016
212 <1> ERR_MISC equ 27 ; miscellaneous/other errors
213 <1> ; 15/10/2016
214 <1> ; TRDOS 8086 -> TRDOS 386 (0Bh -> 28)
215 <1> ERR_INV_FORMAT equ 28 ; 'invalid format !' error
216 <1> ; TRDOS 8086 -> TRDOS 386 (0Dh -> 29)
217 <1> ERR_INV_DATA equ 29 ; 'invalid data !' error
218 <1> ; TRDOS 8086 -> TRDOS 386 (0Eh -> 20)
219 <1> ERR_ZERO_LENGTH equ 20 ; 'zero length !' error
220 <1> ; TRDOS 8086 -> TRDOS 386 (15h -> 17, 1Dh -> 18, 1Eh -> 17)
221 <1> ERR_DRV_NR_READ equ 17 ; 'drive not ready or read error !'
222 <1> ERR_DRV_NR_WRITE equ 18 ; 'drive not ready or write error !'
223 <1> ; 15/10/2016
224 <1> ERR_INV_PATH_NAME equ 19 ; 'bad path name !' error
225 <1> ERR_BAD_CMD_ARG equ 1 ; 'bad command argument !' ; TRDOS 8086
226 <1> ERR_INV_FNUMBER equ 1 ; 'invalid function number !' ; TRDOS 8086
227 <1> ERR_BIG_FILE equ 8 ; 'big file & out of memory !' ; TRDOS 8086
228 <1> ERR_BIG_DATA equ 8 ; 'big data & out of memory !' ; TRDOS 8086
229 <1> ERR_CLUSTER equ 35 ; 'cluster not available !' ; TRDOS 8086
230 <1> ERR_OUT_OF_MEMORY equ 4 ; 'out of memory !'
231 <1> ; 'insufficient memory !'
232 <1> ERR_P_VIOLATION equ 6 ; 'protection violation !'
233 <1> ERR_PAGE_FAULT equ 224 ; 'page fault !' ; 0E0h
234 <1> ERR_SWP_DISK_READ equ 40
235 <1> ERR_SWP_DISK_NOT_PRESENT equ 41
236 <1> ERR_SWP_SECTOR_NOT_PRESENT equ 42
237 <1> ERR_SWP_NO_FREE_SPACE equ 43
238 <1> ERR_SWP_DISK_WRITE equ 44
239 <1> ERR_SWP_NO_PAGE_TO_SWAP equ 45
240 <1> ; 10/04/2017
241 <1> ERR_BUFFER equ 46 ; 'buffer error !'
242 <1> ; 28/08/2017 (20/08/2017)
243 <1> ERR_DMA equ -1 ; DMA buffer (allocation/misc.) error!
244 <1>
245 <1> ; 26/08/2015
246 <1> ; 24/07/2015
247 <1> ; 24/06/2015
248 <1> MAX_ARG_LEN equ 256 ; max. length of sys exec arguments
249 <1> ; 01/07/2015

```

```

250 <1> MAX_MSG_LEN equ 255 ; max. msg length for 'sysmsg'
251 <1> ;
252 <1> ; 06/10/2016
253 <1> ; OPENFILES equ 10 ; max. number of open files (system)
254 <1> ; 23/07/2022
255 <1> OPENFILES equ 32 ; max. number of open files (system)
256 <1> ; 07/10/2016
257 <1> ; NUMOFDEVICES equ 20 ; max. num of available devices (sys)
3431 <1> %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 08h
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> DPDisks 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 0FFFFh
179 <1> FAT16EOC equ 0FFFFh
180 <1> ;FAT32EOC equ 0FFFFFFFh ; It is not direct usable for 8086 code
181 <1> ; BAD Cluster
182 <1> FAT12BADC equ 0FF7h
183 <1> FAT16BADC equ 0FFF7h
184 <1> ;FAT32BADC equ 0FFFFFF7h ; It is not direct usable for 8086 code
185 <1> ; MS-DOS FAT16 FS (Maximum Possible) Last Cluster Number= 0FFF6h
186 <1>
187 <1> ; TRFS
188 <1>
189 <1> bs_FS_JmpBoot equ 0 ; jmp short bsBootCode
190 <1> ; db 0EBh, db 3Fh, db 90h
191 <1> bs_FS_Identifier equ 3 ; db 'FS', db 0
192 <1> bs_FS_BytesPerSec equ 6 ; dw 512
193 <1> bs_FS_MediaAttrib equ 8 ; db 3
194 <1> bs_FS_PartitionID equ 9 ; db 0A1h
195 <1> bs_FS_VersionMaj equ 10 ; db 01h
196 <1> bs_FS_VersionMin equ 11 ; db 0
197 <1> bs_FS_BeginSector equ 12 ; dd 0
198 <1> bs_FS_VolumeSize equ 16 ; dd 2880
199 <1> bs_FS_StartupFD equ 20 ; dd 0
200 <1> bs_FS_MATLocation equ 24 ; dd 1
201 <1> bs_FS_RootDirD equ 28 ; dd 8
202 <1> bs_FS_SystemConfFD equ 32 ; dd 0
203 <1> bs_FS_SwapFD equ 36 ; dd 0
204 <1> bs_FS_UnDelDirD equ 40 ; dd 0
205 <1> bs_FS_DriveNumber equ 44 ; db 0
206 <1> bs_FS_LBA_Ready equ 45 ; db 0
207 <1> bs_FS_MagicWord equ 46
208 <1> bs_FS_SecPerTrack equ 46 ; db 0A1h
209 <1> bs_FS_Heads equ 47 ; db 01h
210 <1> bs_FS_OperationSys equ 48 ; db "TR-SINGLIX v1.0b"
211 <1> bs_FS_Terminator equ 64 ; db 0
212 <1> bs_FS_BootCode equ 65
213 <1>
214 <1> FS_MAT_DATLocation equ 12
215 <1> FS_MAT_DATScount equ 16
216 <1> FS_MAT_FreeSectors equ 20
217 <1> FS_MAT_FirstFreeSector equ 24
218 <1> FS_RDT_VolumeSerialNo equ 28
219 <1> FS_RDT_VolumeName equ 64
220 <1>
221 <1> ; FAT12 + FAT16 + FAT32
222 <1> BS_JmpBoot equ 0
223 <1> BS_OEMName equ 3
224 <1> BPB_BytsPerSec equ 11
225 <1> BPB_SecPerClust equ 13
226 <1> BPB_RsvdSecCnt equ 14
227 <1> BPB_NumFATS equ 16
228 <1> BPB_RootEntCnt equ 17
229 <1> BPB_TotalSec16 equ 19
230 <1> BPB_Media equ 21
231 <1> BPB_FATSz16 equ 22
232 <1> BPB_SecPerTrk equ 24
233 <1> BPB_NumHeads equ 26
234 <1> BPB_HiddSec equ 28
235 <1> BPB_TotalSec32 equ 32
236 <1>
237 <1> ; FAT12 and FAT16 only
238 <1> BS_DrvNum equ 36
239 <1> BS_Reserved1 equ 37

```

```

240 <1> BS_BootSig equ 38
241 <1> BS_VolID equ 39
242 <1> BS_VolLab equ 43
243 <1> BS_FilSysType equ 54 ; 8 bytes
244 <1> BS_BootCode equ 62
245 <1>
246 <1> ; FAT32 only
247 <1> BPB_FATSz32 equ 36 ; FAT32, 4 bytes
248 <1> BPB_ExtFlags equ 40 ; FAT32, 2 bytes
249 <1> BPB_FSVer equ 42 ; FAT32, 2 bytes
250 <1> BPB_RootClus equ 44 ; FAT32, 4 bytes
251 <1> BPB_FSInfo equ 48 ; FAT 32, 2 bytes
252 <1> BPB_BkBootSec equ 50 ; FAT32, 2 bytes
253 <1> BPB_Reserved equ 52 ; FAT32, 12 bytes
254 <1> BS_FAT32_DrvNum equ 64 ; FAT32, 1 byte
255 <1> BS_FAT32_Reserved1 equ 65 ; FAT32, 1 byte
256 <1> BS_FAT32_BootSig equ 66 ; FAT32, 1 byte
257 <1> BS_FAT32_VolID equ 67 ; FAT32, 4 bytes
258 <1> BS_FAT32_VolLab equ 71 ; FAT32, 11 bytes
259 <1> BS_FAT32_FilSysType equ 82 ; FAT32, 8 bytes
260 <1> BS_FAT32_BootCode equ 90
261 <1>
262 <1> ; 29/02/2016
263 <1> ; (FAT32 Free Cluster Count & First Free Cluster values)
264 <1> ; [BPB_Reserved] = Free Cluster Count (offset 52)
265 <1> ; [BPB_Reserved+4] = First Free Cluster (offset 56)
266 <1>
267 <1> BS_Validation equ 510
268 <1>
269 <1> ; 15/02/2016
270 <1> ; FILE.ASM - 09/10/2011
271 <1> ; Directory Entry Structure
272 <1> ; 29/10/2009 (According to Microsoft FAT32 File system Specification)
273 <1> DirEntry_Name equ 0
274 <1> DirEntry_Attr equ 11
275 <1> DirEntry_NTRes equ 12
276 <1> DirEntry_CrtTimeTenth equ 13
277 <1> DirEntry_CrtTime equ 14
278 <1> DirEntry_CrtDate equ 16
279 <1> DirEntry_LastAccDate equ 18
280 <1> DirEntry_FstClusHI equ 20
281 <1> DirEntry_WrtTime equ 22
282 <1> DirEntry_WrtDate equ 24
283 <1> DirEntry_FstClusLO equ 26
284 <1> DirEntry_FileSize equ 28
3432 <1> %include 'trdosk1.s' ; 04/01/2016
1 <1> ; *****
2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.9) - SYS INIT : trdosk1.s
3 <1> ;
4 <1> ; Last Update: 26/09/2024 (Previous: 25/07/2022)
5 <1> ;
6 <1> ; Beginning: 04/01/2016
7 <1> ;
8 <1> ; Assembler: NASM version 2.15 (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> ; 26/09/2024 (TRDOS v2.0.9)
18 <1> ; 18/04/2021 (TRDOS 386 v2.0.4)
19 <1> ; 20/01/2018 (v2.0.1)
20 <1> ; 23/01/2017 (v2.0.0)
21 <1> ; 07/05/2016
22 <1> ; 02/05/2016
23 <1> ; 24/04/2016
24 <1> ; 14/04/2016
25 <1> ; 13/04/2016
26 <1> ; 30/03/2016
27 <1> ; 24/01/2016
28 <1> ; 06/01/2016
29 <1> ; 04/01/2016 (TRDOS 386 v2.0 - Beginning)
30 <1>
31 <1> ; 23/01/2017 - reset timer frequency (to 18.2Hz)
32 <1> mov al, 00110110b ; 36h
33 <1> out 43h, al
34 <1> xor eax, eax ; sub al, al ; 0
35 <1> out 40h, al ; LB
36 <1> out 40h, al ; HB
37 <1>
38 <1> ; 30/03/2016
39 <1> ; Clear Logical DOS Disk Description Tables Area
40 <1> xor eax, eax
41 <1> mov edi, Logical_DOSDisks
42 <1> mov ecx, 6656/4 ; 26*256 = 6656 bytes
43 <1> rep stosd ; 1664 times 4 bytes
44 <1>
45 <1> mov eax, '?:/'
46 <1> mov [Current_Dir_Drv], eax
47 <1>
48 <1> ; Logical DRV INIT (only for hard disks)
49 <1> call ldrv_init ; trdosk2.s
50 <1>
51 <1> ; when floppy_drv_init call is disabled
52 <1> ; media changed sign is needed
53 <1> ; for proper drive initialization
54 <1>
55 <1> mov esi, Logical_DOSDisks
56 <1> mov al, 1 ; Initialization sign (invalid_fd_parameter)
57 <1> add esi, LD_MediaChanged ; Media Change Status = 1 (init needed)
58 <1> mov [esi], al ; A:
59 <1> ; Temporary - 26/09/2024
60 <1> ; mov dword [esi+LD_Clusters], -1 ; *
61 <1> ; dec dword [esi+LD_Clusters]
62 <1> add esi, 100h
63 <1> mov [esi], al ; B:
64 <1> ; mov dword [esi+LD_Clusters], -1 ; *
65 <1> ; dec dword [esi+LD_Clusters]
66 <1>
67 <1> _current_drive_bootdisk:
68 <1> mov dl, [boot_drv] ; physical drive number
69 <1> cmp dl, 0FFh
70 <1> je short _last_dos_diskno_check
71 <1> _boot_drive_check:
72 <1> cmp dl, 80h
73 <1> jb short _current_drive_a
74 <1> sub dl, 7Eh ; C = 2 , D = 3
75 <1> jmp short _current_drive_a
76 <1>
77 <1> _last_dos_diskno_check:
78 <1> mov dl, [Last_DOS_DiskNo]

```

```

79 00006CDE 80FA02    <1>    cmp     dl, 2
80 00006CE1 7706    <1>    ja      short _current_drive_c
81 00006CE3 7406    <1>    je      short _current_drive_a
82 00006CE5 30D2    <1>    xor     dl, dl ; A:
83 00006CE7 EB02    <1>    jmp     short _current_drive_a
84
85    <1>    _current_drive_c:
86 00006CE9 B202    <1>    mov     dl, 2 ; C:
87
88    <1>    _current_drive_a:
89 00006CEB 8815[79660000] <1>    mov     [drv], dl
90 00006CF1 BE[7F2E0100] <1>    mov     esi, msg_CRLF_temp
91 00006CF6 E8AF000000 <1>    call    print_msg
92
93 00006CFB 8A15[79660000] <1>    mov     dl, [drv]
94    <1>    _default_drive_c:
95 00006D01 E8DA090000 <1>    call    change_current_drive
96 00006D06 731C    <1>    jnc     short _start_mainprog
97
98    <1>    _drv_not_ready_error:
99 00006D08 BE[43310100] <1>    mov     esi, msgl_drv_not_ready
100 00006D0D E898000000 <1>    call    print_msg
101    <1>    ; jmp     _end_of_mainprog
102    <1>
103    <1>    ; 20/01/2018
104 00006D12 B202    <1>    mov     dl, 2
105 00006D14 3815[79660000] <1>    cmp     [drv], dl
106 00006D1A 736C    <1>    jnb     short _end_of_mainprog
107 00006D1C 8815[79660000] <1>    mov     [drv], dl
108 00006D22 EBDD    <1>    jmp     short _default_drive_c
109
110    <1>    _start_mainprog:
111    <1>    ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
112    <1>    ; 18/04/2021 (TRDOS 386 v2.0.4 - Beginning)
113    <1>    ; 07/01/2017
114    <1>    ; 07/05/2016
115    <1>    ; 02/05/2016
116    <1>    ; 24/04/2016 (TRDOS 386 v2)
117    <1>    ; Retro UNIX 386 v1, 'sys_init' (u0.s)
118    <1>    ; 23/06/2015
119
120    <1>    ; 02/05/2016
121    <1>    ; 24/04/2016
122    <1>    ; mov     ax, 1
123    <1>    ; 18/04/2021 - TRDOS 386 v2.0.4
124 00006D24 31C0    <1>    xor     eax, eax
125 00006D26 FEC0    <1>    inc     al ; eax = 1
126 00006D28 A2[A58E0100] <1>    mov     [u.uno], al
127 00006D2D 66A3[3C8E0100] <1>    mov     [mpid], ax
128 00006D33 66A3[288D0100] <1>    mov     [p.pid], ax
129 00006D39 A2[888D0100] <1>    mov     [p.stat], al
130 00006D3E C605[9C8E0100]04 <1>    mov     byte [u.quant], time_count ; 07/01/2017
131
132 00006D45 A1[A0760100] <1>    mov     eax, [k_page_dir]
133 00006D4A A3[AC8E0100] <1>    mov     [u.pgdir], eax ; reset
134
135 00006D4F E8D2EAF000 <1>    call    allocate_page
136    <1>    ; jc      panic
137    <1>    ; 25/07/2022
138 00006D54 7305    <1>    jnc     short _start_mainprog_1
139 00006D56 E9AE000000 <1>    jmp     panic
140
141    <1>    _start_mainprog_1:
142 00006D5B A3[A88E0100] <1>    mov     [u.upage], eax ; user structure page
143 00006D60 A3[988D0100] <1>    mov     [p.upage], eax
144 00006D65 E82DEB0000 <1>    call    clear_page
145
146    <1>    ; 24/08/2015
147 00006D6A FE0D[488E0100] <1>    dec     byte [sysflg] ; FFh = ready for system call
148    <1>    ; 0 = executing a system call
149
150    <1>    ; 13/04/2016
151    <1>    ; Clear Environment Variables Page/Area
152 00006D70 BF00300900 <1>    mov     edi, Env_Page ; 93000h
153 00006D75 B980000000 <1>    mov     ecx, Env_Page_Size / 4; 512/4 (4096/4)
154 00006D7A 31C0    <1>    xor     eax, eax
155 00006D7C F3AB    <1>    rep     stosd
156
157 00006D7E E880300000 <1>    call    mainprog_startup_configuration
158
159 00006D83 E8870A0000 <1>    call    dos_prompt
160
161    <1>    _end_of_mainprog:
162 00006D88 BE[7F2E0100] <1>    mov     esi, msg_CRLF_temp
163 00006D8D E818000000 <1>    call    print_msg
164 00006D92 BE[852E0100] <1>    mov     esi, mainprog_Version
165 00006D97 E80E000000 <1>    call    print_msg
166    <1>    ; 24/01/2016
167 00006D9C 28E4    <1>    sub     ah, ah
168 00006D9E E879A1FFFF <1>    call    int16h ; call getch
169 00006DA3 E907A6FFFF <1>    jmp     cpu_reset
170
171 00006DA8 EBFE    <1>    infinitiveloop: jmp short infinitiveloop
172
173    <1>    print_msg:
174    <1>    ; 13/05/2016
175    <1>    ; 04/01/2016
176    <1>    ; 01/07/2015
177    <1>    ; 13/03/2015 (Retro UNIX 386 v1)
178    <1>    ; 07/03/2014 (Retro UNIX 8086 v1)
179    <1>    ; (Modified registers: EAX, EBX, ECX, EDX, ESI, EDI)
180    <1>    ;
181 00006DAA 8A3D[CE760100] <1>    mov     bh, [ACTIVE_PAGE] ; 04/01/2016 (ptty)
182    <1>    ; mov     bl, 07h ; Black background, light gray forecolor
183
184 00006DB0 AC    <1>    lodsb
185
186 00006DB1 56    <1>    pmsg1:
187    <1>    push     esi
188    <1>    ; mov     bh, [ACTIVE_PAGE] ; 04/01/2016 (ptty)
189    <1>    ; mov     bl, 07h ; Black background, light gray forecolor
190 00006DB9 5E    <1>    call    _write_tty
191 00006DBA AC    <1>    pop      esi
192 00006DBB 20C0    <1>    lodsb
193 00006DBD 75F2    <1>    and     al, al
194 00006DBF C3    <1>    jnz     short pmsg1
195    <1>    retn
196
197    <1>    clear_screen:
198    <1>    ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
199    <1>    ; 06/12/2020
200    <1>    ; 03/12/2020 (TRDOS 386 v2.0.3)
201    <1>    ; 13/05/2016
202    <1>    ; 30/01/2016
203    <1>    ; 24/01/2016

```



```

203                                     <1> ; 04/01/2016
204 00006DC0 0FB61D[CE760100] <1> movzx ebx, byte [ACTIVE_PAGE] ; video page number (0 to 7)
205                                     <1> ; 25/07/2022
206                                     <1> ;sub    al, al
207                                     <1> ; al = 0
208 00006DC7 8AA3[2F680000] <1> mov    ah, [ebx+vmode] ; default = 03h (80x25 text)
209 00006DCD 80FC04 <1> cmp    ah, 4
210 00006DD0 7205 <1> jnb    short cls1
211 00006DD2 80FC07 <1> cmp    ah, 7
212 00006DD5 752B <1> jne    short vga_clear
213                                     <1> cls1:
214                                     <1> ;mov    bh, bl
215                                     <1> ;mov    bl, 7
216 00006DD7 3A25[1E680000] <1> cmp    ah, [CRT_MODE] ; current video mode ?
217 00006DDD 740D <1> je     short cls2 ; yes (current video mode = 3)
218                                     <1> ;call   set_mode_3 ; set video mode to 3 (& clear screen)
219                                     <1> ;retn
220                                     <1> ; 06/12/2020
221                                     <1> ;cmp    byte [pmi32], 0
222                                     <1> ; 25/07/2022
223 00006DDF 383D[0C9D0100] <1> cmp    [pmi32], bh ; 0
224 00006DE5 771B <1> ja     short vga_clear
225 00006DE7 E9F9B4FFFF <1> jmp     set_mode_3
226                                     <1> cls2:
227 00006DEC 88DF <1> mov    bh, bl ; video page (0 to 7)
228 00006DEE B307 <1> mov    bl, 07h ; attribute to be used on blanked line
229                                     <1> ; 25/07/2022
230                                     <1> ;sub    al, al ; 0 = entire window
231                                     <1> ;xor    cx, cx
232                                     <1> ; 25/07/2022
233                                     <1> ; al = 0
234 00006DF0 31C9 <1> xor    ecx, ecx
235 00006DF2 66BA4F18 <1> mov    dx, 184Fh
236 00006DF6 E832B2FFFF <1> call   _scroll_up ; 24/01/2016
237                                     <1> ;
238                                     <1> ;mov    bh, [ACTIVE_PAGE] ; video page number (0 to 7)
239                                     <1> ;xor    dx, dx
240                                     <1> ; 25/07/2022
241 00006DFB 31D2 <1> xor    edx, edx
242                                     <1> ;call   _set_cpos ; 24/01/2016
243                                     <1> ;retn
244                                     <1> ; 03/12/2020
245 00006DFD E971B5FFFF <1> jmp     _set_cpos ; returns to the caller of this proc
246                                     <1> ;cls3:
247                                     <1> ; retn
248                                     <1> ;
249                                     <1> ; 06/12/2020
250                                     <1> vga_clear:
251                                     <1> ; 03/12/2020
252                                     <1> ; set mode by using _int10h
253                                     <1> ; (also clears screen)
254                                     <1> ;mov    al, ah
255                                     <1> ;sub    ah, ah ; set current video mode
256                                     <1> ; 25/07/2022
257 00006E02 86C4 <1> xchg   ah, al
258                                     <1> ; ah = 0
259                                     <1> ; al = video mode
260                                     <1> ;call   _int10h ; simulates int 10h in TRDOS 386 kernel
261                                     <1> ;jmp     short cls3
262 00006E04 E91BA9FFFF <1> jmp     _int10h ; returns to the caller of this proc
263                                     <1> ;
264                                     <1> panic:
265                                     <1> ; 13/05/2016 (TRDOS 386 = TRDOS v2)
266                                     <1> ; 13/03/2015 (Retro UNIX 386 v1)
267                                     <1> ; 07/03/2014 (Retro UNIX 8086 v1)
268 00006E09 8E[2F390100] <1> mov    esi, panic_msg
269 00006E0E E897FFFFFF <1> call   print_msg
270                                     <1> key_to_reboot:
271                                     <1> ; 24/01/2016
272 00006E13 28E4 <1> sub    ah, ah
273 00006E15 E802A1FFFF <1> call   int16h ; call getch
274                                     <1> ; wait for a character from the current tty
275                                     <1> ;
276 00006E1A B00A <1> mov    al, 0Ah
277 00006E1C 8A3D[CE760100] <1> mov    bh, [ptty] ; [ACTIVE_PAGE]
278 00006E22 B307 <1> mov    bl, 07h ; Black background,
279                                     <1> ; light gray forecolor
280 00006E24 E8B2B4FFFF <1> call   _write_tty
281 00006E29 E981A5FFFF <1> jmp     cpu_reset
282                                     <1> ;
283                                     <1> ctrlbrk:
284                                     <1> ; 21/08/2024 (TRDOS 386 kernel v2.0.9)
285                                     <1> ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
286                                     <1> ; 12/11/2015
287                                     <1> ; 13/03/2015 (Retro UNIX 386 v1)
288                                     <1> ; 06/12/2013 (Retro UNIX 8086 v1)
289                                     <1> ;
290                                     <1> ; INT 1Bh (control+break) handler
291                                     <1> ;
292                                     <1> ;
293                                     <1> ; Retro Unix 8086 v1 feature only!
294                                     <1> ;
295                                     <1> ; 25/07/2022
296 00006E2E 52 <1> push   edx
297 00006E2F 31D2 <1> xor    edx, edx
298                                     <1> ;
299                                     <1> ; 21/08/2024 - TRDOS 386 v2.0.9
300                                     <1> %if 0
301                                     <1> ;cmp    word [u.intr], 0
302                                     <1> ;cmp    [u.intr], dx ; 0
303                                     <1> ;jna     short cbrk4
304                                     <1> cbrk0:
305                                     <1> ; 12/11/2015
306                                     <1> ; 06/12/2013
307                                     <1> ;cmp    word [u.quit], 0
308                                     <1> ;cmp    [u.quit], dx ; 0 ; 25/07/2022
309                                     <1> ;jz     short cbrk4
310                                     <1> %endif
311                                     <1> ; 20/09/2013
312                                     <1> ;push   ax
313 00006E31 50 <1> push   eax ; 25/07/2022
314 00006E32 A0[CE760100] <1> mov    al, [ptty]
315                                     <1> ;
316                                     <1> ; 21/08/2024 - TRDOS 386 v2.0.9
317                                     <1> %if 1
318 00006E37 3A05[8A8E0100] <1> cmp    al, [u.ttyn]
319 00006E3D 7509 <1> jne    short cbrk3
320                                     <1> %else
321                                     <1> ; 12/11/2015
322                                     <1> ;
323                                     <1> ; ctrl+break (EOT, CTRL+D) from serial port
324                                     <1> ; or ctrl+break from console (pseudo) tty
325                                     <1> ; (!redirection!)
326                                     <1> ;

```

```

327      <1>      cmp     al, 8 ; serial port tty nums > 7
328      <1>      jb      short cbrk1 ; console (pseudo) tty
329      <1>
330      <1>      ; Serial port interrupt handler sets [ptty]
331      <1>      ; to the port's tty number (as temporary).
332      <1>      ;
333      <1>      ; If active process is using a stdin or
334      <1>      ; stdout redirection (by the shell),
335      <1>      ; console tty keyboard must be available
336      <1>      ; to terminate running process,
337      <1>      ; in order to prevent a deadlock.
338      <1>
339      <1>      ; 25/07/2022
340      <1>      ;push     edx
341      <1>      ;movzx    edx, byte [u.uno]
342      <1>      mov     dl, [u.uno]
343      <1>      cmp     al, [edx+p.ttyc-1] ; console tty (rw)
344      <1>      ;pop     edx
345      <1>      je      short cbrk2
346      <1> cbrk1:
347      <1>      inc     al ; [u.ttyp] : 1 based tty number
348      <1>      ; 06/12/2013
349      <1>      cmp     al, [u.ttyp] ; recent open tty (r)
350      <1>      je      short cbrk2
351      <1>      cmp     al, [u.ttyp+1] ; recent open tty (w)
352      <1>      jne     short cbrk3
353      <1> cbrk2:
354      <1>      ; 06/12/2013
355      <1>      ;mov     ax, [u.quit]
356      <1>      ;and     ax, ax
357      <1>      ;jz      short cbrk3
358      <1> %endif
359      <1>      ;xor     ax, ax ; 0
360      <1>      ;dec     ax
361      <1>      ; 0FFFFh = 'ctrl+brk' keystroke
362      <1>      ; 25/07/2022
363      <1>      ;xor     eax, eax ; 0
364      <1>      ;dec     eax ; -1 ; 0FFFFFFFh
365      <1>      ;mov     [u.quit], ax
366      <1>      ; 21/08/2024
367      <1>      ; set CTRL+BREAK flag (even if it is not activated)
368      <1>      ; (u.intr is it's activation flag, 0 = disabled))
369 00006E3F 66C705[A28E0100]FF- <1>      mov     word [u.quit], -1 ; 0FFFFh
369 00006E47 FF <1>
370      <1> cbrk3:
371      <1>      ;pop     ax
372 00006E48 58 <1>      pop     eax ; 25/07/2022
373      <1> cbrk4:
374      <1>      ; 25/07/2022
375 00006E49 5A <1>      pop     edx
376 00006E4A C3 <1>      retn
377      <1>
378      <1> ; 31/12/2017
379      <1> ; TRDOS 386 - 30/12/2017
380      <1> %define get_rtc_date RTC_40
381      <1> %define get_rtc_time RTC_20
382      <1> %define set_rtc_date RTC_50
383      <1> %define set_rtc_time RTC_30
384      <1> get_rtc_date_time:
385      <1> ; Retro UNIX 8086 v1 - UNIX.ASM (01/09/2014)
386      <1> ;epoch:
387      <1>      ; 18/04/2021 (TRDOS 386 v2.0.3)
388      <1>      ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
389      <1>      ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
390      <1>      ; 09/04/2013 (Retro UNIX 8086 v1 - UNIX.ASM)
391      <1>      ; 'epoch' procedure prototype:
392      <1>      ; UNIXCOPY.ASM, 10/03/2013
393      <1>      ; 14/11/2012
394      <1>      ; unixboot.asm (boot file configuration)
395      <1>      ; version of "epoch" procedure in "unixproc.asm"
396      <1>      ; 21/7/2012
397      <1>      ; 15/7/2012
398      <1>      ; 14/7/2012
399      <1>      ; Erdogan Tan - RETRO UNIX v0.1
400      <1>      ; compute current date and time as UNIX Epoch/Time
401      <1>      ; UNIX Epoch: seconds since 1/1/1970 00:00:00
402      <1>      ;
403      <1>      ; ((Modified registers: EAX, EDX, ECX, EBX))
404      <1>      ;
405      <1>
406      <1>      ; 18/04/2021
407      <1>      ; INPUT:
408      <1>      ; none (real time clock)
409      <1>      ; OUTPUT:
410      <1>      ;     eax = unix epoch time value
411      <1>      ;     (seconds since 1/1/1970 00:00:00)
412      <1>
413 00006E4B E801F5FFFF <1>      call    get_rtc_time ; Return Current Time
414      <1>      ;xchg     ch, cl ; 18/04/2021
415 00006E50 66890D[6A730100] <1>      mov     [hour], cx ; BCD, cl = minute, ch = hour
416      <1>      ;xchg     dh, dl ; 18/04/2021
417      <1>      ;mov     [second], dx ; BCD, dh = second, dl = dse
418      <1>      ; 18/04/2021
419 00006E57 8835[6E730100] <1>      mov     [second], dh ; second
420      <1>      ;
421 00006E5D E85AF5FFFF <1>      call    get_rtc_date ; Return Current Date
422      <1>      ;xchg     ch, cl ; 18/04/2021
423 00006E62 66890D[64730100] <1>      mov     [year], cx ; BCD, cl = year, ch = century
424      <1>      ;xchg     dh, dl ; 18/04/2021
425 00006E69 668915[66730100] <1>      mov     [month], dx ; BCD, dl = day, dh = month
426      <1>      ;
427      <1>      ;mov     al, [hour] ; Hour
428      <1>      ; 18/04/2021
429 00006E70 A0[6B730100] <1>      mov     al, [hour+1] ; Hour
430      <1>      ; AL <-- BCD number
431 00006E75 D410 <1>      db      0D4h, 10h ; Undocumented inst. AAM
432      <1>      ; AH = AL / 10h
433      <1>      ; AL = AL MOD 10h
434 00006E77 D50A <1>      aad ; AX= AH*10+AL
435      <1>      ;mov     [hour], al
436      <1>      ;mov     al, [hour+1] ; Minute
437 00006E79 8605[6A730100] <1>      xchg     al, [hour] ; [hour] = hour, al = minute
438      <1>      ; AL <-- BCD number
439 00006E7F D410 <1>      db      0D4h, 10h ; Undocumented inst. AAM
440      <1>      ; AH = AL / 10h
441      <1>      ; AL = AL MOD 10h
442 00006E81 D50A <1>      aad ; AX= AH*10+AL
443 00006E83 A2[6C730100] <1>      mov     [minute], al
444 00006E88 A0[6E730100] <1>      mov     al, [second] ; Second
445      <1>      ; AL <-- BCD number
446 00006E8D D410 <1>      db      0D4h, 10h ; Undocumented inst. AAM
447      <1>      ; AH = AL / 10h
448      <1>      ; AL = AL MOD 10h
449 00006E8F D50A <1>      aad ; AX= AH*10+AL

```

```

450 00006E91 A2[6E730100] <1> mov [second], al
451 00006E96 66A1[64730100] <1> mov ax, [year] ; Year (century)
452 <1> ; 18/04/2021
453 <1> ;push eax ; puhs ax
454 <1> ;mov al, ah ; century ; 18/04/2021
455 <1> ; AL <-- BCD number
456 00006E9C D410 <1> db 0D4h, 10h ; Undocumented inst. AAM
457 <1> ; AH = AL / 10h
458 <1> ; AL = AL MOD 10h
459 00006E9E D50A <1> aad ; AX= AH*10+AL
460 <1> ;mov ah, 100
461 <1> ;mul ah
462 <1> ;mov [year], ax
463 <1> ; 18/04/2021
464 <1> ; ax = al = year (0 to 99)
465 00006EA0 668705[64730100] <1> xchg ax, [year] ; [year+1] = century -> ah
466 <1> ;pop eax ; pop ax
467 00006EA7 88E0 <1> mov al, ah ; century
468 <1> ; AL <-- BCD number
469 00006EA9 D410 <1> db 0D4h, 10h ; Undocumented inst. AAM
470 <1> ; AH = AL / 10h
471 <1> ; AL = AL MOD 10h
472 00006EAB D50A <1> aad ; AX= AH*10+AL
473 <1> ; 18/04/2021
474 00006EAD B464 <1> mov ah, 100 ; 100*(century byte of year)
475 00006EAF F6E4 <1> mul ah
476 <1> ;
477 00006EB1 660105[64730100] <1> add [year], ax
478 <1> ;mov al, [month] ; Month
479 <1> ; 18/04/2021
480 00006EB8 A0[67730100] <1> mov al, [month+1] ; Month
481 <1> ; AL <-- BCD number
482 00006EBD D410 <1> db 0D4h, 10h ; Undocumented inst. AAM
483 <1> ; AH = AL / 10h
484 <1> ; AL = AL MOD 10h
485 00006EBF D50A <1> aad ; AX= AH*10+AL
486 <1> ;mov [month], al
487 <1> ;mov al, [month+1] ; Day
488 <1> ; 18/04/2021
489 00006EC1 8605[66730100] <1> xchg al, [month] ; [month] = month, al = day
490 <1> ; AL <-- BCD number
491 00006EC7 D410 <1> db 0D4h, 10h ; Undocumented inst. AAM
492 <1> ; AH = AL / 10h
493 <1> ; AL = AL MOD 10h
494 00006EC9 D50A <1> aad ; AX= AH*10+AL
495 00006ECB A2[68730100] <1> mov [day], al
496 <1>
497 00006ED0 C3 <1> retn ; 30/12/2017
498 <1>
499 <1> epoch:
500 00006ED1 E875FFFFFF <1> call get_rtc_date_time ; TRDOS 386 - 30/12/2017
501 <1>
502 <1> convert_to_epoch:
503 <1> ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
504 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
505 <1> ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit modification)
506 <1> ; 09/04/2013 (Retro UNIX 8086 v1)
507 <1> ;
508 <1> ; ((Modified registers: EAX, EDX, EBX))
509 <1> ;
510 <1> ; Derived from DALLAS Semiconductor
511 <1> ; Application Note 31 (DS1602/DS1603)
512 <1> ; 6 May 1998
513 00006ED6 29C0 <1> sub eax, eax
514 00006ED8 66A1[64730100] <1> mov ax, [year]
515 00006EDE 662DB207 <1> sub ax, 1970
516 00006EE2 BA6D010000 <1> mov edx, 365
517 00006EE7 F7E2 <1> mul edx
518 00006EE9 31DB <1> xor ebx, ebx
519 00006EEB 8A1D[66730100] <1> mov bl, [month]
520 00006EF1 FECB <1> dec bl
521 00006EF3 D0E3 <1> shl bl, 1
522 <1> ;sub edx, edx
523 00006EF5 668B93[70730100] <1> mov dx, [EBX+DMonth]
524 00006EFC 8A1D[68730100] <1> mov bl, [day]
525 00006F02 FECB <1> dec bl
526 00006F04 01D0 <1> add eax, edx
527 00006F06 01D8 <1> add eax, ebx
528 <1> ; EAX = days since 1/1/1970
529 00006F08 668B15[64730100] <1> mov dx, [year]
530 00006F0F 6681EAB107 <1> sub dx, 1969
531 <1> ;shr dx, 1
532 <1> ;shr dx, 1
533 <1> ; 25/07/2022
534 00006F14 C1EA02 <1> shr edx, 2
535 <1> ; (year-1969)/4
536 00006F17 01D0 <1> add eax, edx
537 <1> ; + leap days since 1/1/1970
538 00006F19 803D[66730100]02 <1> cmp byte [month], 2 ; if past february
539 00006F20 760E <1> jna short cte1
540 00006F22 668B15[64730100] <1> mov dx, [year]
541 00006F29 6683E203 <1> and dx, 3 ; year mod 4
542 00006F2D 7501 <1> jnz short cte1
543 <1> ; and if leap year
544 <1> ;add eax, 1 ; add this year's leap day (february 29)
545 <1> ; 25/07/2022
546 00006F2F 40 <1> inc eax
547 <1> cte1: ; compute seconds since 1/1/1970
548 00006F30 BA18000000 <1> mov edx, 24
549 00006F35 F7E2 <1> mul edx
550 00006F37 8A15[6A730100] <1> mov dl, [hour]
551 00006F3D 01D0 <1> add eax, edx
552 <1> ; EAX = hours since 1/1/1970 00:00:00
553 <1> ;mov ebx, 60
554 00006F3F B33C <1> mov bl, 60
555 00006F41 F7E3 <1> mul ebx
556 00006F43 8A15[6C730100] <1> mov dl, [minute]
557 00006F49 01D0 <1> add eax, edx
558 <1> ; EAX = minutes since 1/1/1970 00:00:00
559 <1> ;mov ebx, 60
560 00006F4B F7E3 <1> mul ebx
561 00006F4D 8A15[6E730100] <1> mov dl, [second]
562 00006F53 01D0 <1> add eax, edx
563 <1> ; EAX -> seconds since 1/1/1970 00:00:00
564 00006F55 C3 <1> retn
565 <1>
566 <1> ;set_date_time:
567 <1> convert_from_epoch:
568 <1> ; 25/07/2022 (v2.0.5)
569 <1> ; 18/04/2021 (v2.0.4)
570 <1> ; 31/12/2017 (v2.0.0)
571 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
572 <1> ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
573 <1> ; 20/06/2013 (Retro UNIX 8086 v1)

```

```

574      <1>      ; 'convert_from_epoch' procedure prototype:
575      <1>      ;
576      <1>      ; UNIXCOPY.ASM, 10/03/2013
577      <1>      ;
578      <1>      ; ((Modified registers: EAX, EDX, ECX, EBX))
579      <1>      ;
580      <1>      ; Derived from DALLAS Semiconductor
581      <1>      ; Application Note 31 (DS1602/DS1603)
582      <1>      ; 6 May 1998
583      <1>      ;
584      <1>      ; INPUT:
585      <1>      ; EAX = Unix (Epoch) Time
586      <1>      ;
587      <1>      xor     edx, edx
588      <1>      ;mov     ecx, 60
589      <1>      ; 25/07/2022
590      <1>      sub     ecx, ecx
591      <1>      mov     cl, 60
592      <1>      div     ecx
593      <1>      ;mov     [imin], eax      ; whole minutes
594      <1>      ; since 1/1/1970
595      <1>      mov     [second], dx    ; leftover seconds
596      <1>      sub     edx, edx
597      <1>      div     ecx
598      <1>      ;mov     [ihrs], eax     ; whole hours
599      <1>      ; since 1/1/1970
600      <1>      mov     [minute], dx   ; leftover minutes
601      <1>      xor     edx, edx
602      <1>      ;mov     cx, 24
603      <1>      mov     cl, 24
604      <1>      div     ecx
605      <1>      ;mov     [iday], ax      ; whole days
606      <1>      ; since 1/1/1970
607      <1>      mov     [hour], dx     ; leftover hours
608      <1>      add     eax, 365+366    ; whole day since
609      <1>      ; 1/1/1968
610      <1>      ;mov     [iday], ax
611      <1>      push    eax
612      <1>      sub     edx, edx
613      <1>      ;mov     ecx, (4*365)+1 ; 4 years = 1461 days
614      <1>      ; 25/07/2022
615      <1>      mov     cx, (4*365)+1
616      <1>      div     ecx
617      <1>      pop     ecx
618      <1>      ;mov     [lday], ax     ; count of quadyrs (4 years)
619      <1>      ;push    dx
620      <1>      ; 18/04/2021
621      <1>      push    edx
622      <1>      ;mov     [qday], dx     ; days since quadyr began
623      <1>      cmp     dx, 31+29     ; if past feb 29 then
624      <1>      cmc     eax, 0        ; add this quadyr's leap day
625      <1>      ;mov     [lday], ax     ; to # of qadyrs (leap days)
626      <1>      ; since 1968
627      <1>      ;mov     cx, [iday]
628      <1>      xchg    ecx, eax       ; ECX = lday, EAX = iday
629      <1>      sub     eax, ecx       ; iday - lday
630      <1>      ;mov     ecx, 365
631      <1>      ; 25/07/2022
632      <1>      mov     cx, 365
633      <1>      xor     edx, edx
634      <1>      ; EAX = iday-lday, EDX = 0
635      <1>      div     ecx
636      <1>      ;mov     [iyrs], ax     ; whole years since 1968
637      <1>      ;jday = iday - (iyrs*365) - lday
638      <1>      ;mov     [jday], dx     ; days since 1/1 of current year
639      <1>      ;add     eax, 1968
640      <1>      add     ax, 1968       ; compute year
641      <1>      mov     [year], ax
642      <1>      ;mov     cx, dx
643      <1>      ; 25/07/2022
644      <1>      mov     ecx, edx
645      <1>      ;mov     dx, [qday]
646      <1>      ;pop     dx
647      <1>      ; 18/04/2021
648      <1>      pop     edx
649      <1>      cmp     dx, 365         ; if qday <= 365 and qday >= 60
650      <1>      ja     short cfe1     ; jday = jday + 1
651      <1>      cmp     dx, 60         ; if past 2/29 and leap year then
652      <1>      cmc     cx, 0          ; add a leap day to the # of whole
653      <1>      ; 25/07/2022           ; days since 1/1 of current year
654      <1>      adc     ecx, 0
655      <1>      cfe1:
656      <1>      ;mov     [jday], cx
657      <1>      ;mov     bx, 12         ; estimate month
658      <1>      ; 18/04/2021
659      <1>      sub     ebx, ebx
660      <1>      mov     bl, 12
661      <1>      mov     dx, 366         ; mday, max. days since 1/1 is 365
662      <1>      and     ax, 11b        ; year mod 4 (and dx, 3)
663      <1>      cfe2:
664      <1>      ; Month calculation ; 0 to 11 (11 to 0)
665      <1>      cmp     cx, dx         ; mday = # of days passed from 1/1
666      <1>      ; 25/07/2022
667      <1>      cmp     ecx, edx
668      <1>      jnb     short cfe3
669      <1>      dec     bx             ; month = month - 1
670      <1>      shl     bx, 1
671      <1>      ; 18/04/2021
672      <1>      dec     bl
673      <1>      shl     bl, 1
674      <1>      mov     dx, [EBX+DMonth] ; # elapsed days at 1st of month
675      <1>      ; 18/04/2021
676      <1>      shr     bx, 1         ; bx = month - 1 (0 to 11)
677      <1>      shr     bl, 1
678      <1>      cmp     bx, 1         ; if month > 2 and year mod 4 = 0
679      <1>      cmp     bl, 1
680      <1>      jna     short cfe2     ; then mday = mday + 1
681      <1>      jna     short cfe2     ; then mday = mday + 1
682      <1>      or      al, al         ; if past 2/29 and leap year then
683      <1>      jnz     short cfe2     ; add leap day (to mday)
684      <1>      inc     dx             ; mday = mday + 1
685      <1>      ; 25/07/2022
686      <1>      inc     edx
687      <1>      jmp     short cfe2
688      <1>      cfe3:
689      <1>      inc     bx             ; -> bx = month, 1 to 12
690      <1>      ; 18/04/2021
691      <1>      inc     bl
692      <1>      mov     [month], bx
693      <1>      sub     cx, dx         ; day = jday - mday + 1
694      <1>      ; 25/07/2022
695      <1>      sub     ecx, edx
696      <1>      inc     cx
697      <1>      ; 18/04/2021
698      <1>      inc     cl

```

```

698      <1>      ;mov     [day], cx
699 00006FF4 880D[68730100] <1>      mov     [day], cl
700      <1>
701      <1>      ; eax, ebx, ecx, edx is changed at return
702      <1>      ; output ->
703      <1>      ; [year], [month], [day], [hour], [minute], [second]
704      <1>
705 00006FFA C3 <1>      retn     ; 31/12/2017 (TRDOS 386)
706      <1>
707      <1>      set_rtc_date_time:
708      <1>      ; 31/12/2017 (v2.0.0)
709      <1>      ; 30/12/2017 (TRDOS 386)
710      <1>      ; 15/03/2015 (Retro UNIX 386 v1 - 32 bit version)
711      <1>      ; 20/06/2013 (Retro UNIX 8086 v1)
712 00006FFB E80F000000 <1>      call    set_date_bcd
713      <1>      ; Set real-time clock date
714 00007000 E8DDF3FFFF <1>      call    set_rtc_date ; RTC_50
715      <1>      ; Set real-time clock time
716 00007005 E832000000 <1>      call    set_time_bcd
717 0000700A E970F3FFFF <1>      jmp     set_rtc_time ; RTC_30
718      <1>
719      <1>      ; 31/12/2017
720      <1>      set_date_bcd:
721      <1>      mov     al, [year+1]
722 00007014 D40A <1>      aam     ; ah = al / 10, al = al mod 10
723 00007016 D510 <1>      db      0D5h, 10h ; Undocumented inst. AAD
724      <1>      ; AL = AH * 10h + AL
725 00007018 88C5 <1>      mov     ch, al ; century (BCD)
726 0000701A A0[64730100] <1>      mov     al, [year]
727 0000701F D40A <1>      aam     ; ah = al / 10, al = al mod 10
728 00007021 D510 <1>      db      0D5h, 10h ; Undocumented inst. AAD
729      <1>      ; AL = AH * 10h + AL
730 00007023 88C1 <1>      mov     cl, al ; year (BCD)
731 00007025 A0[66730100] <1>      mov     al, [month]
732 0000702A D40A <1>      aam     ; ah = al / 10, al = al mod 10
733 0000702C D510 <1>      db      0D5h, 10h ; Undocumented inst. AAD
734      <1>      ; AL = AH * 10h + AL
735 0000702E 88C6 <1>      mov     dh, al ; month (BCD)
736 00007030 A0[68730100] <1>      mov     al, [day]
737 00007035 D40A <1>      aam     ; ah = al / 10, al = al mod 10
738 00007037 D510 <1>      db      0D5h, 10h ; Undocumented inst. AAD
739      <1>      ; AL = AH * 10h + AL
740      <1>      ; 18/04/2021
741 00007039 88C2 <1>      mov     dl, al ; day (BCD)
742 0000703B C3 <1>      retn     ; 30/12/2017
743      <1>
744      <1>      ; 31/12/2017
745      <1>      set_time_bcd:
746      <1>      ; Read real-time clock time
747      <1>      ; (get day light saving time bit status)
748 0000703C FA <1>      cli
749 0000703D E8DBF4FFFF <1>      call    UPD_IPR ; CHECK FOR UPDATE IN PROCESS
750      <1>      ; cf = 1 -> al = 0
751 00007042 7207 <1>      jc      short stime1
752 00007044 B00B <1>      mov     al, CMOS_REG_B ; ADDRESS ALARM REGISTER
753 00007046 E808F5FFFF <1>      call    CMOS_READ ; READ CURRENT VALUE OF DSE BIT
754      <1>      stime1:
755 0000704B FB <1>      sti
756 0000704C 2401 <1>      and     al, 00000001b ; MASK FOR VALID DSE BIT
757 0000704E 88C2 <1>      mov     dl, al ; SET [DL] TO ZERO FOR NO DSE BIT
758      <1>      ; DL = 1 or 0 (day light saving time)
759      <1>
760 00007050 A0[6A730100] <1>      mov     al, [hour]
761 00007055 D40A <1>      aam     ; ah = al / 10, al = al mod 10
762 00007057 D510 <1>      db      0D5h, 10h ; Undocumented inst. AAD
763      <1>      ; AL = AH * 10h + AL
764 00007059 88C5 <1>      mov     ch, al ; hour (BCD)
765 0000705B A0[6C730100] <1>      mov     al, [minute]
766 00007060 D40A <1>      aam     ; ah = al / 10, al = al mod 10
767 00007062 D510 <1>      db      0D5h, 10h ; Undocumented inst. AAD
768      <1>      ; AL = AH * 10h + AL
769 00007064 88C1 <1>      mov     cl, al ; minute (BCD)
770 00007066 A0[6E730100] <1>      mov     al, [second]
771 0000706B D40A <1>      aam     ; ah = al / 10, al = al mod 10
772 0000706D D510 <1>      db      0D5h, 10h ; Undocumented inst. AAD
773      <1>      ; AL = AH * 10h + AL
774 0000706F 88C6 <1>      mov     dh, al ; second (BCD)
775 00007071 C3 <1>      retn     ; 30/12/2017
3433      <1>      %include 'trdosk2.s' ; 04/01/2016
1      <1>      ; *****
2      <1>      ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.10) - DRV INIT : trdosk2.s
3      <1>      ;
4      <1>      ; -----
5      <1>      ; Last Update: 19/12/2025 (Previous: 22/05/2024, TRDOS 386 v2.0.8)
6      <1>      ; -----
7      <1>      ; Beginning: 04/01/2016
8      <1>      ; -----
9      <1>      ; Assembler: NASM version 2.14 (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 (c) 2009-2011 Erdogan TAN [26/09/2009] Last Update: 07/08/2011
15     <1>      ;
16     <1>      ; ldrv_init: ; Logical Drive Initialization
17     <1>      ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
18     <1>      ; 22/05/2024 (TRDOS 386 kernel v2.0.8)
19     <1>      ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
20     <1>      ; 30/08/2020
21     <1>      ; 25/08/2020
22     <1>      ; 11/08/2020 - 13/08/2020
23     <1>      ; 17/07/2020 - 20/07/2020
24     <1>      ; 14/07/2020 - 15/07/2020
25     <1>      ; 30/01/2018
26     <1>      ; 27/12/2017
27     <1>      ; 12/02/2016
28     <1>      ; 06/01/2016
29     <1>      ; ('diskinit.inc', 'diskio.inc' integration)
30     <1>      ; 04/01/2016 (TRDOS 386 = TRDOS v2.0)
31     <1>      ; 07/08/2011
32     <1>      ; 20/09/2009
33     <1>      ; 2005
34     <1>
35     <1>      ; 15/07/2020
36     <1>      ; movzx ecx, byte [HF_NUM] ; number of fixed disks
37     <1>      ; cmp cl, 1
38     <1>      ; jnb short load_hd_partition_tables
39     <1>
40 00007072 A0[38770100] <1>      mov     al, [HF_NUM] ; number of fixed disks
41 00007077 20C0 <1>      and     al, al
42 00007079 7501 <1>      jnz     short load_hd_partition_tables
43     <1>
44     <1>      ; no any hard disks
45 0000707B C3 <1>      retn

```

```

46      <1>
47      <1> load_hd_partition_tables:
48      <1>     ;mov     esi, [HDPM_TBL_VEC] ; primary master disk FDPT
49      <1>     ; 15/07/2020
50 0000707C BE[3C770100] <1>     mov     esi, HDPM_TBL_VEC
51 00007081 BF[627B0100] <1>     mov     edi, PTable_hd0
52 00007086 B280 <1>     mov     dl, 80h
53 <1>     ; 15/07/2020
54 00007088 A2[7B660000] <1>     mov     [hdc], al
55 <1>     ;xor     ecx, ecx ; 0
56 <1> load_next_hd_partition_table:
57 <1>     ; 20/07/2020
58 0000708D 31C9 <1>     xor     ecx, ecx ; 0
59 <1>     ;push    ecx
60 0000708F 57 <1>     push    edi ; *
61 <1>     ;push    esi ; FDPT (+ DPTE) address
62 <1>     ; 15/07/2020
63 00007090 AD <1>     lodsd
64 00007091 56 <1>     push    esi ; ** ; next FDPT (+ DPTE) address ptr
65 <1>
66 <1>     ;mov     al, [esi+20] ; DPTE offset 4
67 <1>     ;and     al, 40h ; LBA bit (bit 6)
68 <1>     ;shr     al, 6
69 <1>     ;mov     [HD_LBA_yes], al
70 <1>
71 <1>     ; 15/07/2020
72 00007092 8A4814 <1>     mov     cl, [eax+20]
73 00007095 80E140 <1>     and     cl, 40h
74 <1>     ;mov     [HD_LBA_yes], cl
75 <1>     ; 22/05/2024 (BugFix)
76 00007098 0FB6C2 <1>     movzx   eax, dl
77 0000709B 05[E27B0100] <1>     add     eax, HD_LBA_yes - 80h
78 000070A0 8808 <1>     mov     [eax], cl
79 <1>
80 000070A2 E81A030000 <1>     call    load_masterboot
81 <1>     ;jc     short pass_pt_this_hard_disk
82 <1>     ; 13/08/2020
83 <1>     ;jc     pass_pt_this_hard_disk
84 <1>     ; 25/07/2022
85 000070A7 7302 <1>     jnc     short load_mbr_ok
86 000070A9 EB7C <1>     jmp     pass_pt_this_hard_disk
87 <1>
88 <1> load_mbr_ok:
89 000070AB BB[207B0100] <1>     mov     ebx, PartitionTable
90 000070B0 89DE <1>     mov     esi, ebx
91 <1>     ;mov     ecx, 16
92 000070B2 B110 <1>     mov     cl, 16
93 000070B4 F3A5 <1>     rep     movsd
94 000070B6 89DE <1>     mov     esi, ebx
95 <1>     ;mov     byte [hdc], 4 ; 4 - partition index
96 <1>     ; 15/07/2020
97 000070B8 C605[677C0100]04 <1>     mov     byte [PP_Counter], 4
98 <1> loc_validate_hdp_partition:
99 <1>     ;cmp     byte [esi+ptFileSystemID], 0
100 <1>     ;jna     short loc_validate_next_hdp_partition2
101 <1>     ; 13/08/2020
102 000070BF 8A4604 <1>     mov     al, [esi+ptFileSystemID]
103 000070C2 20C0 <1>     and     al, al
104 000070C4 7449 <1>     jz      short loc_validate_next_hdp_partition2
105 <1>
106 000070C6 56 <1>     push    esi ; *** ; Masterboot partition table offset
107 000070C7 52 <1>     push    edx ; **** ; dl = Physical drive number
108 <1>
109 <1>     ; 13/08/2020
110 000070C8 3C05 <1>     cmp     al, 05h ; Extended partition CHS
111 000070CA 7404 <1>     je      short loc_set_ep_counter
112 000070CC 3C0F <1>     cmp     al, 0Fh ; Extended partition LBA
113 000070CE 7511 <1>     jne     short loc_validate_next_hdp_partition0
114 <1>
115 <1>     ;inc     byte [PP_Counter]
116 <1>     ; 15/07/2020
117 <1>     ;inc     byte [EP_Counter] ; disk has valid partition(s)
118 <1>
119 <1> loc_set_ep_counter:
120 <1>     ; 13/08/2020
121 000070D0 803D[687C0100]80 <1>     cmp     byte [EP_Counter], 80h
122 000070D7 7334 <1>     jnb     short loc_validate_next_hdp_partition1
123 <1>
124 000070D9 8815[687C0100] <1>     mov     byte [EP_Counter], dl ; disk drv has extd. part.
125 <1>
126 000070DF EB2C <1>     jmp     short loc_validate_next_hdp_partition1
127 <1>
128 <1> loc_validate_next_hdp_partition0:
129 000070E1 31FF <1>     xor     edi, edi ; 0
130 <1>     ; Input -> ESI = PartitionTable offset
131 <1>     ; DL = Hard disk drive number
132 <1>     ; EDI = 0 -> Primary Partition
133 <1>     ; EDI > 0 -> Extended Partition's Start Sector
134 000070E3 E87E010000 <1>     call    validate_hd_fat_partition
135 <1>
136 <1>     ; 19/12/2025
137 <1>     %if 0
138 <1>     jnc     short loc_set_valid_hdp_partition_entry
139 <1>
140 <1>     ;pop     edx
141 <1>     ;push    edx
142 <1>     mov     edx, [esp] ; ****
143 <1>     mov     esi, [esp+4] ; *** ; 30/01/2018
144 <1>     call    validate_hd_fs_partition
145 <1>     %endif
146 000070E8 7223 <1>     jc      short loc_validate_next_hdp_partition1
147 <1>
148 <1> loc_set_valid_hdp_partition_entry:
149 000070EA 8A0D[7D2E0100] <1>     mov     cl, [Last_DOS_DiskNo]
150 000070F0 80C141 <1>     add     cl, 'A'
151 <1>     ; ESI = Logical dos drive description table address
152 000070F3 880E <1>     mov     [esi+LD_Name], cl
153 <1>     ; 15/07/2020
154 000070F5 8A4602 <1>     mov     al, [esi+LD_PhyDrvNo] ; Physical drive number
155 <1>     ;mov     al, [esp] ; ****
156 000070F8 2C7F <1>     sub     al, 7Fh
157 <1>     ; AL = 1 to 4
158 000070FA C0E002 <1>     shl     al, 2 ; AL = 4 to 16
159 <1>
160 000070FD 8A15[677C0100] <1>     mov     dl, [PP_Counter]
161 <1>
162 <1>     ;sub     al, [PP_Counter]
163 00007103 28D0 <1>     sub     al, dl ; [PP_Counter] ; 4 - partition index
164 <1>
165 <1>     ; AL = Partition entry/index, 0 based
166 <1>     ; 0 -> hd 0, Partition Table offset = 0
167 <1>     ; 15 -> hd 3, Partition Table offset = 3
168 <1>
169 <1>     ;mov     [esi+LD_PartitionEntry], al

```

```

170 <1>
171 <1> ; 15/07/2020
172 00007105 B404 <1> mov ah, 4
173 <1> ;sub ah, [PP_Counter]
174 00007107 28D4 <1> sub ah, dl
175 <1>
176 <1> ; AH = Primary partition index, 0 to 3 ; pt entry
177 <1> ; (4 to 7 for logical disk partitions)
178 <1>
179 <1> ;mov [esi+LD_DParamEntry], ah
180 00007109 6689467C <1> mov [esi+LD_PartitionEntry], ax
181 <1>
182 <1> loc_validate_next_hdp_partition1:
183 0000710D 5A <1> pop edx ; **** ; dl = Physical drive number
184 0000710E 5E <1> pop esi ; *** ; Masterboot partition table offset
185 <1>
186 <1> loc_validate_next_hdp_partition2:
187 <1> ; ESI = PartitionTable offset
188 <1> ; DL = Hard/Fixed disk drive number
189 <1>
190 <1> ;dec byte [hdc] ; 4 - partition index
191 <1> ;jz short pass_pt_this_hard_disk
192 <1> ; 15/07/2020
193 0000710F FE0D[677C0100] <1> dec byte [PP_Counter] ; 4 - partition index
194 00007115 7410 <1> jz short pass_pt_this_hard_disk
195 <1>
196 00007117 83C610 <1> add esi, 16 ; 10h
197 0000711A EBA3 <1> jmp short loc_validate_hdp_partition
198 <1>
199 <1> loc_not_any_extd_partitions:
200 <1> ; 15/07/2020
201 0000711C C3 <1> retn
202 <1>
203 <1> loc_next_hd_partition_table:
204 0000711D FEC2 <1> inc dl
205 <1> ; 15/07/2020
206 <1> ;add esi, 32 ; next FDPT address
207 0000711F 83C740 <1> add edi, 64 ; next partition table destination
208 00007122 E966FFFFFF <1> jmp load_next_hd_partition_table
209 <1>
210 <1> pass_pt_this_hard_disk:
211 <1> ;pop esi ; FDPT (+ DPTE) address
212 <1> ; 15/07/2020
213 00007127 5E <1> pop esi ; ** ; next FDPT (+ DPTE) address ptr
214 00007128 5F <1> pop edi ; * ; Ptable_hd?
215 <1> ;pop ecx
216 <1> ;loop loc_next_hd_partition_table
217 00007129 FE0D[7B660000] <1> dec byte [hdc]
218 0000712F 75EC <1> jnz short loc_next_hd_partition_table
219 <1>
220 <1> ;cmp byte [PP_Counter], 1
221 <1> ;jnb short load_extended_dos_partitions
222 <1> ; Empty partition table
223 <1> ;retn
224 <1>
225 <1> ; 11/08/2020
226 <1> ; 17/07/2020
227 <1> check_extended_partitions:
228 <1> ; 15/07/2020
229 <1> ;cmp byte [EP_Counter], 0
230 <1> ;jna short loc_not_any_extd_partitions
231 <1> ; 13/08/2020
232 00007131 A0[687C0100] <1> mov al, [EP_Counter] ; 1st disk drv has extd partition
233 00007136 08C0 <1> or al, al ; 0 ?
234 00007138 74E2 <1> jz short loc_not_any_extd_partitions
235 <1>
236 <1> load_extended_dos_partitions:
237 <1> ;mov byte [hdc], 80h
238 <1> ; 13/08/2020
239 0000713A A2[7B660000] <1> mov byte [hdc], al ; 1st disk drv has extd partition
240 <1> ; 25/08/2020
241 0000713F 2C80 <1> sub al, 80h
242 00007141 740E <1> jz short loc_set_ext_ptable_hd0
243 00007143 C0E006 <1> shl al, 6 ; * 64
244 00007146 0FB6F0 <1> movzx esi, al
245 00007149 81C6[627B0100] <1> add esi, PTable_hd0
246 0000714F EB05 <1> jmp short next_hd_extd_partition
247 <1>
248 <1> ; 25/08/2020
249 <1> loc_set_ext_ptable_hd0:
250 00007151 BE[627B0100] <1> mov esi, PTable_hd0
251 <1>
252 <1> next_hd_extd_partition:
253 <1> ; 17/07/2020
254 <1> ;mov byte [EP_Counter], 0 ; Reset for each physical disk
255 <1> ; 13/08/2020
256 <1> ;mov byte [LD_Counter], 0 ; Reset logical drive index
257 00007156 66C705[687C0100]00- <1> mov word [EP_Counter], 0 ; Reset EP index and LD index
258 0000715E 00 <1>
259 <1>
260 0000715F 56 <1> push esi ; **** ; PTable_hd? offset
261 <1>
262 00007160 C605[677C0100]04 <1> mov byte [PP_Counter], 4
263 <1> ; set for each extd partition table
264 <1> ;mov ecx, 4
265 00007167 8A15[7B660000] <1> mov cl, 4
266 <1> mov dl, [hdc]
267 0000716D 8A4604 <1> hd_check_fs_id_05h:
268 00007170 3C05 <1> mov al, [esi+ptFileSystemID]
269 00007172 7411 <1> cmp al, 05h ; Is it an extended dos partition ?
270 <1> je short loc_set_ep_start_sector ; yes
271 00007174 3C0F <1> hd_check_fs_id_0Fh:
272 00007176 740D <1> cmp al, 0Fh ; Is it an extended win4 (LBA mode) partition ?
273 <1> je short loc_set_ep_start_sector ; yes
274 <1>
275 <1> continue_to_check_ep:
276 <1> ;add esi, 16
277 <1> ;loop hd_check_fs_id_05h
278 <1> ; 15/07/2020
279 <1> ;dec cl
280 00007178 FE0D[677C0100] <1> ;jz short continue_check_ep_next_disk
281 0000717E 7432 <1> dec byte [PP_Counter] ; 4 --> 0
282 00007180 83C610 <1> jz short continue_check_ep_next_disk
283 00007183 EBE8 <1> add esi, 16
284 <1> jmp short hd_check_fs_id_05h
285 <1>
286 <1> loc_set_ep_start_sector:
287 <1> ; dl = [hdc] ; Drive number
288 <1> ; 15/07/2020
289 00007185 8B4E08 <1> mov ecx, [esi+ptStartSector]
290 00007188 890D[6A7C0100] <1> ; 30/08/2020
291 <1> mov [MBR_EP_StartSector], ecx
292 <1> ; 20/07/2020
292 <1> loc_validate_hde_partition_next:

```

```

293      <1> ; 22/05/2024 (BugFix)
294 0000718E 0FB6FA <1> movzx edi, dl
295 00007191 81C7[E27B0100] <1> add edi, HD_LBA_yes - 80h
296      <1> ;
297 00007197 890D[6E7C0100] <1> mov [EP_StartSector], ecx ; Extended partition's start sector
298 0000719D BB[62790100] <1> mov ebx, MasterBootBuff
299      <1> ; 22/05/2024
300 000071A2 803F01 <1> cmp byte [edi], 1 ; LBA ready = Yes
301      <1> ; cmp byte [HD_LBA_yes], 1 ; LBA ready = Yes
302 000071A5 7227 <1> jnb short loc_hd_load_ep_05h ; cf = 1 ; 20/07/2020
303      <1> ; 11/08/2020
304      <1> ; (BugFix for extended partition type 05h beyond CHS limit)
305      <1> ; (Infact if extended partition starts at the beyond of CHS limit,
306      <1> ; it's partition ID must be 0Fh but they/somebodies had used 05h.)
307      <1> ; cmp al, 05h
308      <1> ; je short loc_hd_load_ep_05h
309      <1> loc_hd_load_ep_0Fh:
310      <1> ; 04/01/2016
311      <1> ; push ecx
312      <1> ; 15/07/2020
313      <1> ; mov ecx, [esi+ptStartSector] ; sector number
314      <1> ; mov ebx, MasterBootBuff ; buffer address
315      <1> ; LBA read/write (with private LBA function)
316      <1> ; ((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
317      <1> ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
318      <1> ; mov ah, 1Bh ; LBA read
319      <1> ; mov al, 1 ; sector count
320 000071A7 66B8011B <1> mov ax, 1B01h
321 000071AB E85BDEFFFF <1> call int13h
322      <1> ; pop ecx
323      <1> ; jnc short loc_hd_move_ep_table
324      <1> ; 15/07/2020
325 000071B0 732E <1> jnc short loc_validate_hde_partition
326      <1> ;
327      <1> continue_check_ep_next_disk:
328      <1> ; 15/07/2020
329      <1> ; pop edi ; PTable_ep?
330 000071B2 5E <1> pop esi ; **** ; PTable_hd?
331 000071B3 A0[38770100] <1> mov al, [HF_NUM] ; number of hard disks
332 000071B8 047F <1> add al, 7Fh
333 000071BA 3805[7B660000] <1> cmp [hdc], al
334 000071C0 730B <1> jnb short loc_validating_hd_partitions_ok
335 000071C2 83C640 <1> add esi, 64
336      <1> ; 15/07/2020
337      <1> ; add edi, 64
338 000071C5 FE05[7B660000] <1> inc byte [hdc]
339 000071CB EB89 <1> jmp short next_hd_extd_partition
340      <1> ;
341      <1> loc_validating_hd_partitions_ok:
342      <1> ; 15/07/2020
343      <1> ; mov al, [Last_DOS_DiskNo]
344      <1> loc_drv_init_retn:
345 000071CD C3 <1> retn
346      <1> ;
347      <1> loc_hd_load_ep_05h:
348      <1> ; 20/07/2020 ('diskio.s', int13h, cf = 1 -> bugfix)
349      <1> ; cld ; (Bug: int13h would not clear carry flag bit,
350      <1> ; ; even if there would not be an error)
351      <1> ; ; ((Fix: now, int13h procedure clears carry flag
352      <1> ; ; at the entrance of it.. 20/07/2020))
353      <1> ; 15/07/2020
354      <1> ; push ecx
355 000071CE 8A7601 <1> mov dh, [esi+ptBeginHead]
356 000071D1 668B4E02 <1> mov cx, [esi+ptBeginSector]
357 000071D5 66B80102 <1> mov ax, 0201h ; Read 1 sector
358      <1> ; mov ebx, MasterBootBuff
359 000071D9 E82DDEFFFF <1> call int13h ; 20/07/2020
360      <1> ; ; 'diskio.s' modification, 'cld'
361      <1> ; pop ecx
362 000071DE 72D2 <1> jc short continue_check_ep_next_disk
363      <1> ; 15/07/2020
364      <1> ; jmp short loc_validate_hde_partition
365      <1> ;
366      <1> ; 15/07/2020
367      <1> ; loc_hd_move_ep_table:
368      <1> ; ; pop edi
369      <1> ; ; push edi ; PTable_ep?
370      <1> ; mov edi, [esp]
371      <1> ; mov esi, PartitionTable ; Extended
372      <1> ; mov ebx, esi
373      <1> ; mov ecx, 16
374      <1> ; mov cl, 16
375      <1> ; rep movsd
376      <1> ; mov esi, ebx
377      <1> ; loc_set_hde_sub_partition_count:
378      <1> ; mov byte [PP_Counter], 4
379      <1> ; mov byte [EP_Counter], 0
380      <1> ;
381      <1> loc_validate_hde_partition:
382      <1> ; 13/08/2020
383      <1> ; 15/07/2020
384      <1> ; mov byte [PP_Counter], 4
385 000071E0 BE[207B0100] <1> mov esi, PartitionTable ; (in MasterBootBuff)
386      <1> ; 13/08/2020
387      <1> ; jmp short get_minidisk_partition_entry
388      <1> ;
389      <1> ; get_minidisk_partition_entry:
390      <1> ; ; 20/07/2020
391      <1> ; cmp byte [esi+ptFileSystemID], 0
392      <1> ; ja short loc_validate_minidisk_partition
393      <1> ; ; 13/08/2020
394      <1> ; jmp short continue_check_ep_next_disk
395      <1> ;
396      <1> ; ; 11/08/2020
397      <1> ; get_minidisk_partition_entry_next:
398      <1> ; ; 13/08/2020
399      <1> ; ; dec byte [PP_Counter]
400      <1> ; ; jz short continue_check_ep_next_disk
401      <1> ; ; 20/07/2020
402      <1> ; ; get_minidisk_partition_entry_next:
403      <1> ; ; 13/08/2020
404      <1> ; ; cmp esi, PartitionTable+64
405      <1> ; ; jnb short continue_check_ep_next_disk
406      <1> ; ;
407      <1> ; ; add esi, 16 ; 10h
408      <1> ; ; jmp short get_minidisk_partition_entry
409      <1> ;
410      <1> ; 13/08/2020
411      <1> ; get_minidisk_partition_entry:
412      <1> ; 20/07/2020
413 000071E5 807E0400 <1> cmp byte [esi+ptFileSystemID], 0
414 000071E9 76C7 <1> jna short continue_check_ep_next_disk ; 13/08/2020
415      <1> ;
416      <1> loc_validate_minidisk_partition:

```



```

417      <1>      ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
418      <1>      ; 13/08/2020
419      <1>      ; 20/07/2020
420      <1>      ;push esi ; *** ; Extended partition table offset
421      <1>
422      <1>      ; 13/08/2020
423 000071EB FE05[687C0100] <1>      inc byte [EP_Counter] ; current (sub partition) index
424      <1>      ; in current extended partition
425 000071F1 BF[6E7C0100] <1>      mov edi, EP_StartSector
426      <1>
427      <1>      ; Input -> ESI = PartitionTable offset
428      <1>      ; DL = Hard disk drive number
429      <1>      ; EDI = Extended partition start sector pointer
430 000071F6 E86B000000 <1>      call validate_hd_fat_partition
431      <1>      ;pop ecx ; *
432 000071FB 7308 <1>      jnc short loc_set_valid_hde_partition_entry
433      <1>      ; jump down to deep !!!
434      <1>
435      <1>      ;pop esi ; *** ; Extended partition table offset
436      <1>      ; 13/08/2020
437      <1>      ;mov esi, PartitionTable
438      <1>
439      <1>      ; 11/08/2020
440      <1>      ; ESI = Extended partition table offset
441 000071FD 8A15[7B660000] <1>      mov dl, [hdc]
442      <1>
443      <1>      ;; DL = Hard disk drive number
444      <1>      ;dec byte [PP_Counter]
445      <1>      ;jz short continue_check_ep_next_disk
446      <1>      ;add esi, 16 ; 10h
447      <1>      ;mov dl, [hdc]
448      <1>      ;jmp short get_minidisk_partition_entry
449      <1>
450      <1>      ; 11/08/2020
451      <1>      ;jmp short get_minidisk_partition_entry_next
452      <1>
453      <1>      ; 23/08/2020
454 00007203 EB3E <1>      jmp short validate_next_minidisk_partition_ok
455      <1>
456      <1>      ; 17/07/2020
457      <1>      ;; jumping down to deep levels !!!
458      <1>      ; ((That is a pitty microsoft preferred ep table chain
459      <1>      ; instead of a single table as mbr partition table!??))
460      <1>
461      <1>      loc_set_valid_hde_partition_entry:
462      <1>      ; 15/07/2020
463 00007205 A0[7B660000] <1>      mov al, [hdc] ; Hard disk drive number (>=80h)
464 0000720A 88C2 <1>      mov dl, al ; mov dl, [hdc]
465 0000720C 2C7F <1>      sub al, 7Fh
466      <1>      ; 1 to 4
467 0000720E C0E002 <1>      shl al, 2 ; 4 to 16
468 00007211 2A05[677C0100] <1>      sub al, [PP_Counter] ; al - (4 - partition index)
469      <1>      ; (disk number * 4) + partition index
470      <1>
471      <1>      ; AL = Partition entry/index, 0 based
472      <1>      ; 0 -> hd 0, Partition Table offset = 0
473      <1>      ; 15 -> hd 3, Partition Table offset = 3
474      <1>
475      <1>      ;mov ah, 4 ; Logical dos partition (>= 4)
476      <1>      ;add ah, [EP_Counter]
477      <1>      ; Logical disk partition index = 4 to 7
478      <1>      ; (Primary disk partition index = 0 to 3)
479      <1>
480      <1>      ; 13/08/2020
481 00007217 8A25[697C0100] <1>      mov ah, [LD_Counter] ; Logical drive index number
482      <1>      ; (in current extended partition)
483 0000721D 80C404 <1>      add ah, 4 ; 4 to 7
484      <1>
485      <1>      ; 15/07/2020
486      <1>      ; CX -> AX
487      <1>      ;; 06/01/2016 (TRDOS v2.0)
488      <1>      ;; BUGFIX *
489      <1>      ;;mov [esi+LD_PartitionEntry], cl
490      <1>      ;;mov [esi+LD_DParamEntry], ch
491      <1>      ;mov [esi+LD_PartitionEntry], cx
492 00007220 6689467C <1>      mov [esi+LD_PartitionEntry], ax
493      <1>
494 00007224 8A0D[7D2E0100] <1>      mov cl, [Last_DOS_DiskNo]
495 0000722A 80C141 <1>      add cl, 'A'
496 0000722D 880E <1>      mov [esi+LD_Name], cl
497      <1>
498      <1>      ; 17/07/2020
499      <1>      ;cmp cl, 'z'
500      <1>      ;jb short logical_drive_count_ok_for_next
501      <1>      ;pop esi ; ***
502      <1>      ;pop esi ; ****
503      <1>      ;retn
504      <1>
505      <1>      ;logical_drive_count_ok_for_next:
506      <1>
507      <1>      ;; 15/07/2020
508      <1>      ;inc byte [EP_Counter]
509      <1>      ; 13/08/2020
510 0000722F FE05[697C0100] <1>      inc byte [LD_Counter]
511      <1>
512      <1>      ;mov dl, [hdc]
513      <1>
514      <1>      ; 17/07/2020
515      <1>      ;; Now,
516      <1>      ;; we are swimming in deep of an extended partition !!!
517      <1>      ; (! sub or chained extended partition tables !)
518      <1>      ; ((Logical dos partitions in extended partition were called
519      <1>      ; as 'mini disk partition' in msdos 6.0 source code.))
520      <1>
521      <1>      validate_next_minidisk_partition:
522      <1>      ; 13/08/2020
523      <1>      ;pop esi ; *** ; Extended partition table offset
524      <1>
525      <1>      ; 17/07/2020
526      <1>      ;cmp byte [EP_Counter], 4
527      <1>      ; 13/08/2020
528 00007235 803D[697C0100]04 <1>      cmp byte [LD_Counter], 4 ; maximum 4 logical disks
529      <1>      ; per extended partition
530      <1>      ;jnb continue_check_ep_next_disk
531      <1>      ; 25/07/2022
532 0000723C 7205 <1>      jb short validate_next_minidisk_partition_ok
533 0000723E E96FFFFFFF <1>      jmp continue_check_ep_next_disk
534      <1>
535      <1>      validate_next_minidisk_partition_ok:
536      <1>      ; 13/08/2020
537      <1>      ;dec byte [PP_Counter] ; 4 --> 0
538      <1>      ;jz continue_check_ep_next_disk
539      <1>
540      <1>      ;cmp esi, PartitionTable+64

```

```

541      <1>      ;jnb      continue_check_ep_next_disk
542      <1>
543      <1>      ;add      esi, 16
544      <1>      ; 13/08/2020
545 00007243 BE[307B0100] <1>      mov      esi, PartitionTable+16
546      <1>
547      <1>      ; 20/07/2020
548 00007248 8A4604 <1>      mov      al, [esi+ptFileSystemID]
549      <1>
550      <1>      ; 20/07/2020
551 0000724B 3C05 <1>      cmp      al, 05h ; Is it an extended dos partition ?
552 0000724D 7409 <1>      je       short loc_minidisk_next_ep_lba_chs ; 17/07/2020
553 0000724F 3C0F <1>      cmp      al, 0Fh ; Is it an extended win4 (LBA mode) partition ?
554      <1>      ;jne      continue_check_ep_next_disk ; AL must be 0 here
555      <1>      ; (when it is not 05h or 0Fh)
556      <1>      ; If AL is not ZERO -> EP Bug!
557      <1>      ; (!Microsoft DOS convention!)
558      <1>      ; 25/07/2022
559 00007251 7405 <1>      je       short loc_minidisk_next_ep_lba_chs
560 00007253 E95AFFFFFF <1>      jmp      continue_check_ep_next_disk
561      <1>
562      <1>      loc_minidisk_next_ep_lba_chs:
563      <1>      ; 17/07/2020
564 00007258 8B4E08 <1>      mov      ecx, [esi+ptStartSector] ; relative start sector number
565      <1>      ;add      ecx, [EP_StartSector]
566      <1>      ; 30/08/2020
567 0000725B 030D[6A7C0100] <1>      add      ecx, [MBR_EP_StartSector]
568      <1>      ; 20/07/2020
569 00007261 E928FFFFFF <1>      jmp      loc_validate_hde_partition_next
570      <1>
571      <1>      validate_hd_fat_partition:
572      <1>      ; 17/07/2020
573      <1>      ; 15/07/2020
574      <1>      ; (optimization)
575      <1>      ; 14/07/2020
576      <1>      ; (fat16 -big- partition search bugfix)
577      <1>      ; 27/12/2017
578      <1>      ; 12/02/2016
579      <1>      ; 07/01/2016 (TRDOS 386 = TRDOS v2.0)
580      <1>      ; 07/08/2011
581      <1>      ; 23/07/2011
582      <1>      ; Input
583      <1>      ; DL = Hard/Fixed Disk Drive Number
584      <1>      ; ESI = PartitionTable offset
585      <1>      ; EDI = Extend. Part. Start Sector Pointer
586      <1>      ; EDI = 0 -> Primary Partition
587      <1>      ; byte [Last_DOS_DiskNo]
588      <1>      ; Output
589      <1>      ; cf=0 -> Validated
590      <1>      ; ESI = Logical dos drv desc. table
591      <1>      ; EBX = FAT boot sector buffer
592      <1>      ; byte [Last_DOS_DiskNo]
593      <1>      ; cf=1 -> Not a valid FAT partition
594      <1>      ; EAX, EDX, ECX, EDI -> changed
595      <1>
596      <1>      ;mov      esi, PartitionTable
597 00007266 8A6604 <1>      mov      ah, [esi+ptFileSystemID]
598 00007269 8002 <1>      mov      al, 2 ; 27/12/2017
599 0000726B 80FC06 <1>      cmp      ah, 06h ; FAT16 CHS partition (>=32MB)
600      <1>      ; 12/02/2016
601      <1>      ; ;jb      short loc_not_a_valid_fat_partition2
602      <1>      ; jnb      short vhdh_FAT16_32
603      <1>      ; 14/07/2020 (BugFix)
604 0000726E 7711 <1>      ja       short vhdh_FAT16_32
605 00007270 7425 <1>      je       short loc_set_valid_hd_partition_params
606      <1>
607      <1>      vhdh_FAT12_16:
608      <1>      ; 27/12/2017
609 00007272 FEC8 <1>      dec      al ; mov al, 1
610 00007274 38C4 <1>      cmp      ah, al ; 1 ; FAT12 partition
611 00007276 741F <1>      je       short loc_set_valid_hd_partition_params
612      <1>      ;
613 00007278 FEC0 <1>      inc      al ; mov al, 2
614 0000727A 80FC04 <1>      cmp      ah, 04h ; FAT16 CHS partition (< 32MB)
615 0000727D 7418 <1>      je       short loc_set_valid_hd_partition_params
616      <1>
617      <1>      ; 15/07/2020
618      <1>      ; (ah = 05h, 02h or 03h)
619      <1>      loc_not_a_valid_fat_partition1:
620 0000727F F9 <1>      stc
621      <1>      ; cf=1
622 00007280 C3 <1>      retn
623      <1>
624      <1>      vhdh_FAT16_32:
625      <1>      ; 15/07/2020
626      <1>      ;mov      al, 3
627 00007281 FEC0 <1>      inc      al
628 00007283 80FC0C <1>      cmp      ah, 0Ch ; FAT32 LBA partition
629 00007286 740F <1>      je       short loc_set_valid_hd_partition_params
630 00007288 7706 <1>      ja       short vhdh_check_FAT16_lba
631      <1>
632      <1>      vhdh_check_FAT32_chs:
633 0000728A 80FC0B <1>      cmp      ah, 0Bh ; FAT32 CHS partition
634 0000728D 7408 <1>      je       short loc_set_valid_hd_partition_params
635      <1>      ;jne      short loc_not_a_valid_fat_partition1
636      <1>
637      <1>      ;stc
638      <1>      loc_not_a_valid_fat_partition2:
639 0000728F C3 <1>      retn
640      <1>
641      <1>      vhdh_check_FAT16_lba:
642 00007290 80FC0E <1>      cmp      ah, 0Eh ; FAT16 LBA partition
643 00007293 75EA <1>      jne      short loc_not_a_valid_fat_partition1
644      <1>
645      <1>      ;mov      al, 2
646 00007295 FEC8 <1>      dec      al
647      <1>
648      <1>      loc_set_valid_hd_partition_params:
649      <1>      ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
650      <1>      ; 30/07/2022
651      <1>      ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
652      <1>      ; 15/07/2020
653      <1>      ;inc      byte [Last_DOS_DiskNo] ; > 1
654      <1>      ;
655 00007297 31DB <1>      xor      ebx, ebx
656 00007299 8A3D[7D2E0100] <1>      mov      bh, [Last_DOS_DiskNo] ; * 256
657 0000729F FEC7 <1>      inc      bh ; 15/07/2020
658 000072A1 81C300010900 <1>      add      ebx, Logical_DOSDisks
659      <1>      ;
660 000072A7 C6430102 <1>      mov      byte [ebx+LD_DiskType], 2
661 000072AB 885302 <1>      mov      byte [ebx+LD_PhyDrvNo], dl
662      <1>      ;mov      byte [ebx+LD_FATType], al ; 2 or 3
663      <1>      ;mov      byte [ebx+LD_FSType], ah ; 06h, 0Eh, 0Bh, 0Ch
664 000072AE 66894303 <1>      mov      word [ebx+LD_FATType], ax

```

```

665      ;
666 000072B2 8B4E08      <1>      mov     ecx, [esi+ptStartSector]
667 000072B5 09FF      <1>      or      edi, edi
668 000072B7 7402      <1>      jz      short pass_hd_FAT_ep_start_sector_adding
669      <1> loc_add_hd_FAT_ep_start_sector:
670      <1>      ; 17/07/2020
671 000072B9 030F      <1>      add     ecx, [edi]
672      <1> pass_hd_FAT_ep_start_sector_adding:
673 000072BB 894B6C      <1>      mov     [ebx+LD_StartSector], ecx
674      <1> loc_hd_FAT_logical_drv_init:
675 000072BE 89DD      <1>      mov     ebp, ebx
676      <1>      ;mov     dl, [ebx+LD_PhyDrvNo]
677 000072C0 A0[627C0100] <1>      mov     al, [HD_LBA_yes] ; 07/01/2016
678 000072C5 884305      <1>      mov     [ebx+LD_LBAYes], al
679 000072C8 BB[727C0100] <1>      mov     ebx, DOSBootSectorBuff ; buffer address
680 000072CD 08C0      <1>      or      al, al
681 000072CF 740C      <1>      jz      short loc_hd_FAT_drv_init_load_bs_chs
682      <1> loc_hd_FAT_drv_init_load_bs_lba:
683      <1>      ; DL = Physical drive number
684      <1>      ;mov     ecx, [esi+ptStartSector] ; sector number
685      <1>      ;mov     ebx, DOSBootSectorBuff ; buffer address
686      <1>      ; LBA read/write (with private LBA function)
687      <1>      ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
688      <1>      ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
689 000072D1 B41B      <1>      mov     ah, 1Bh ; LBA read
690 000072D3 B001      <1>      mov     al, 1 ; sector count
691 000072D5 E831DDFFFF <1>      call    int13h
692 000072DA 7313      <1>      jnc     short loc_hd_drv_FAT_boot_validation
693      <1> loc_not_a_valid_fat_partition3:
694 000072DC C3      <1>      retn
695      <1> loc_hd_FAT_drv_init_load_bs_chs:
696 000072DD 8A7601      <1>      mov     dh, [esi+ptBeginHead]
697 000072E0 668B4E02 <1>      mov     cx, [esi+ptBeginSector]
698 000072E4 66B80102 <1>      mov     ax, 0201h ; Read 1 sector
699      <1>      ;mov     ebx, DOSBootSectorBuff
700 000072E8 E81EDDFFFF <1>      call    int13h
701 000072ED 72ED      <1>      jc      short loc_not_a_valid_fat_partition3
702      <1> loc_hd_drv_FAT_boot_validation:
703      <1>      ;mov     esi, DOSBootSectorBuff
704 000072EF 89DE      <1>      mov     esi, ebx
705 000072F1 6681BEFE01000055AA <1>      cmp     word [esi+BS_Validation], 0AA55h
706 000072FA 751C      <1>      jne     short loc_not_a_valid_fat_partition4
707 000072FC 807E15F8 <1>      cmp     byte [esi+BPB_Media], 0F8h
708 00007300 7516      <1>      jne     short loc_not_a_valid_fat_partition4
709      <1>      ; 19/12/2025
710 00007302 66817E0B0002 <1>      cmp     word [esi+BPB_BytsPerSec], 512
711 00007308 750E      <1>      jne     short loc_not_a_valid_fat_partition4
712      <1>
713      <1>      ; 25/07/2022
714 0000730A 31C9      <1>      xor     ecx, ecx
715      <1>
716      <1>      ; 27/12/2017
717 0000730C 807D0303 <1>      cmp     byte [ebp+LD_FATType], 3
718 00007310 7508      <1>      jne     short loc_hd_FAT16_BPB
719      <1>
720      <1> loc_hd_drv_FAT32_boot_validation:
721 00007312 807E4229 <1>      cmp     byte [esi+BS_FAT32_BootSig], 29h
722 00007316 7413      <1>      je      short loc_hd_FAT32_BPB
723      <1>
724      <1> loc_not_a_valid_fat_partition4:
725 00007318 F9      <1>      stc
726 00007319 C3      <1>      retn
727      <1>
728      <1> loc_hd_FAT16_BPB:
729 0000731A 807E2629 <1>      cmp     byte [esi+BS_BootSig], 29h
730 0000731E 75F8      <1>      jne     short loc_not_a_valid_fat_partition4
731      <1>
732 00007320 66837E1600 <1>      cmp     word [esi+BPB_FATSz16], 0
733 00007325 7604      <1>      jna     short loc_hd_big_FAT16_BPB
734      <1>      ;mov     ecx, 32
735      <1>      ; 25/07/2022
736 00007327 B120      <1>      mov     cl, 32
737      <1>      ; ecx = 32
738 00007329 EB02      <1>      jmp     short loc_hd_move_FAT_BPB
739      <1>
740      <1> loc_hd_FAT32_BPB:
741      <1>      ;cmp     word [esi+BPB_FATSz16], 0
742      <1>      ;ja      short loc_not_a_valid_fat_partition4
743      <1> loc_hd_big_FAT16_BPB:
744      <1>      ;mov     ecx, 45
745      <1>      ; 25/07/2022
746 0000732B B12D      <1>      mov     cl, 45
747      <1>      ; ecx = 45
748      <1> loc_hd_move_FAT_BPB:
749 0000732D 89EF      <1>      mov     edi, ebp
750      <1>      ;mov     esi, ebx ; Boot sector
751 0000732F 57      <1>      push    edi
752 00007330 83C706      <1>      add     edi, LD_BPB
753 00007333 F366A5      <1>      rep     movsw
754 00007336 5E      <1>      pop     esi
755 00007337 0FB74614 <1>      movzx   eax, word [esi+LD_BPB+BPB_RsvdSecCnt]
756 0000733B 03466C      <1>      add     eax, [esi+LD_StartSector]
757 0000733E 894660      <1>      mov     [esi+LD_FATBegin], eax
758 00007341 807E0303 <1>      cmp     byte [esi+LD_FATType], 3
759 00007345 7223      <1>      jb      short loc_set_FAT16_RootDirLoc
760      <1> loc_set_FAT32_RootDirLoc:
761 00007347 8B462A      <1>      mov     eax, [esi+LD_BPB+BPB_FATSz32]
762 0000734A 0FB65E16 <1>      movzx   ebx, byte [esi+LD_BPB+BPB_NumFATS]
763 0000734E F7E3      <1>      mul     ebx
764 00007350 034660      <1>      add     eax, [esi+LD_FATBegin]
765      <1> loc_set_FAT32_data_begin:
766 00007353 894668      <1>      mov     [esi+LD_DATABegin], eax
767 00007356 894664      <1>      mov     [esi+LD_ROOTBegin], eax
768      <1>      ; If Root Directory Cluster <> 2 then
769      <1>      ; change the beginning sector value
770      <1>      ; of the root dir by adding sector offset.
771 00007359 8B4632      <1>      mov     eax, [esi+LD_BPB+BPB_RootClus]
772      <1>      ;sub     eax, 2
773      <1>      ; 30/07/2022
774 0000735C 48      <1>      dec     eax ; 2 -> 1
775 0000735D 48      <1>      dec     eax ; 1 -> 0
776 0000735E 7433      <1>      jz      short short loc_set_32bit_FAT_total_sectors
777      <1>      ;movzx   ebx, byte [esi+LD_BPB+BPB_SecPerClust]
778 00007360 8A5E13      <1>      mov     bl, [esi+LD_BPB+BPB_SecPerClust]
779 00007363 F7E3      <1>      mul     ebx
780 00007365 014664      <1>      add     [esi+LD_ROOTBegin], eax
781 00007368 EB29      <1>      jmp     short loc_set_32bit_FAT_total_sectors
782      <1>
783      <1> loc_set_FAT16_RootDirLoc:
784 0000736A 0FB64616 <1>      movzx   eax, byte [esi+LD_BPB+BPB_NumFATS]
785 0000736E 0FB7561C <1>      movzx   edx, word [esi+LD_BPB+BPB_FATSz16]
786 00007372 F7E2      <1>      mul     edx
787 00007374 034660      <1>      add     [esi+LD_FATBegin], eax
788 00007377 894664      <1>      mov     [esi+LD_ROOTBegin], eax

```

```

789 <1> loc_set_FAT16_data_begin:
790 0000737A 894668 <1> mov [esi+LD_DATABegin], eax
791 <1> ;mov eax, 20h ; Size of a directory entry
792 <1> ;movzx edx, word [esi+LD_BPB+BPB_RootEntCnt]
793 <1> ;mov dx, [esi+LD_BPB+BPB_RootEntCnt]
794 <1> ;mul edx
795 <1> ;mov ecx, 511
796 <1> ;mov cx, 511
797 <1> ;add eax, ecx
798 <1> ;inc ecx ; 512
799 <1> ;div ecx
800 <1> ; 14/07/2020
801 0000737D 0FB74617 <1> movzx eax, word [esi+LD_BPB+BPB_RootEntCnt]
802 00007381 6683C00F <1> add ax, 15
803 <1> ;shr ax, 4 ; / 16 ; (16 entries per sector)
804 <1> ; 25/07/2022
805 00007385 C1E804 <1> shr eax, 4
806 00007388 014668 <1> add [esi+LD_DATABegin], eax
807 <1> ;movzx eax, word [esi+LD_BPB+BPB_TotalSec16]
808 0000738B 668B4619 <1> mov ax, [esi+LD_BPB+BPB_TotalSec16]
809 <1> ;test ax, ax
810 <1> ; 25/07/2022
811 0000738F 85C0 <1> test eax, eax
812 <1> ;jz short loc_set_32bit_FAT_total_sectors
813 <1> ;loc_set_16bit_FAT_total_sectors:
814 <1> ;mov [esi+LD_TotalSectors], eax
815 <1> ;jmp short loc_set_hd_FAT_cluster_count
816 <1> ; 14/07/2020
817 00007391 7503 <1> jnz short loc_set_hd_FAT_cluster_count
818 <1> loc_set_32bit_FAT_total_sectors:
819 00007393 8B4626 <1> mov eax, [esi+LD_BPB+BPB_TotalSec32]
820 <1> ;mov [esi+LD_TotalSectors], eax
821 <1> loc_set_hd_FAT_cluster_count:
822 00007396 894670 <1> mov [esi+LD_TotalSectors], eax ; 14/07/2020
823 00007399 8B5668 <1> mov edx, [esi+LD_DATABegin]
824 0000739C 2B566C <1> sub edx, [esi+LD_StartSector]
825 0000739F 29D0 <1> sub eax, edx
826 000073A1 31D2 <1> xor edx, edx ; 0
827 000073A3 0FB64E13 <1> movzx ecx, byte [esi+LD_BPB+BPB_SecPerClust]
828 000073A7 F7F1 <1> div ecx
829 000073A9 894678 <1> mov [esi+LD_Clusters], eax
830 <1> ; Maximum Valid Cluster Number= EAX +1
831 <1> ; with 2 reserved clusters= EAX +2
832 <1> loc_set_hd_FAT_fs_free_sectors:
833 <1> ;mov dword [esi+LD_FreeSectors], 0
834 000073AC E831000000 <1> call get_free_FAT_sectors
835 000073B1 720D <1> jc short loc_validate_hd_FAT_partition_retn
836 000073B3 894674 <1> mov [esi+LD_FreeSectors], eax
837 000073B6 C6467E06 <1> mov byte [esi+LD_MediaChanged], 6 ; Volume Name Reset
838 <1>
839 <1> ; 15/07/2020
840 000073BA FE05[7D2E0100] <1> inc byte [Last_DOS_DiskNo] ; > 1
841 <1>
842 <1> ;mov cl, [Last_DOS_DiskNo]
843 <1> ;add cl, 'A'
844 <1> ;mov [esi+LD_FS_Name], cl
845 <1>
846 <1> loc_validate_hd_FAT_partition_retn:
847 000073C0 C3 <1> retn
848 <1>
849 <1> ; 19/12/2025
850 <1> %if 0
851 <1>
852 <1> validate_hd_fs_partition:
853 <1> ; 19/12/2025 (TRDOS 386 v2.0.10)
854 <1> ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
855 <1> ; 03/02/2018
856 <1> ; 09/12/2017
857 <1> ; 13/02/2016
858 <1> ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
859 <1> ; 29/01/2011
860 <1> ; 23/07/2011
861 <1> ; Input
862 <1> ; DL = Hard/Fixed Disk Drive Number
863 <1> ; ESI = PartitionTable offset
864 <1> ; byte [Last_DOS_DiskNo]
865 <1> ; Output
866 <1> ; cf=0 -> validated
867 <1> ; ESI = Logical dos drv desc. table
868 <1> ; EBX = Singlix FS boot sector buffer
869 <1> ; byte [Last_DOS_DiskNo]
870 <1> ; cf=1 -> Not a valid 'Singlix FS' partition
871 <1> ; EAX, EDX, ECX, EDI -> changed
872 <1>
873 <1> ;mov esi, PartitionTable
874 <1> mov ah, [esi+ptFileSystemID]
875 <1> cmp ah, 0A1h ; SINGLIX FS1 (trfs1) partition
876 <1> jne short loc_validate_hd_fs_partition_stc_retn
877 <1> loc_set_valid_hd_fs_partition_params:
878 <1> inc byte [Last_DOS_DiskNo] ; > 1
879 <1> xor al, al ; mov al, 0
880 <1> ;mov [drv], dl
881 <1> sub ebx, ebx ; 0
882 <1> mov bh, [Last_DOS_DiskNo]
883 <1> add ebx, Logical_DOSDisks
884 <1> mov byte [ebx+LD_DiskType], 2
885 <1> mov [ebx+LD_PhyDrvNo], dl
886 <1> ;mov [ebx+LD_FATType], al ; 0
887 <1> ;mov [ebx+LD_FSType], ah
888 <1> mov [ebx+LD_FATType], ax
889 <1> ;mov eax, [esi+ptStartSector]
890 <1> ;mov [ebx+LD_StartSector], eax
891 <1> loc_hd_fs_logical_drv_init:
892 <1> mov ebp, ebx ; 10/01/2016
893 <1> ;mov dl, [ebx+LD_PhyDrvNo]
894 <1> mov al, [HD_LBA_yes] ; 10/01/2016
895 <1> mov [ebx+LD_LBAYes], al
896 <1> mov esi, ebx
897 <1> mov ebx, DOSBootSectorBuff ; buffer address
898 <1> or al, al
899 <1> jnz short loc_hd_fs_drv_init_load_bs_lba
900 <1> loc_hd_fs_drv_init_load_bs_chs:
901 <1> mov dh, [esi+ptBeginHead]
902 <1> mov cx, [esi+ptBeginSector]
903 <1> mov ax, 0201h ; Read 1 sector
904 <1> ;mov ebx, DOSBootSectorBuff
905 <1> call int13h
906 <1> jnc short loc_hd_drv_fs_boot_validation
907 <1> loc_validate_hd_fs_partition_err_retn:
908 <1> retn
909 <1> loc_validate_hd_fs_partition_stc_retn:
910 <1> stc
911 <1> retn
912 <1> loc_hd_fs_drv_init_load_bs_lba:

```

```

913     <1> ; DL = Physical drive number
914     <1> ;mov     esi, ebx
915     <1> mov     ecx, [esi+ptStartSector] ; sector number
916     <1> ;mov     ebx, DOSBootSectorBuff ; buffer address
917     <1> ; LBA read/write (with private LBA function)
918     <1> ;((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))
919     <1> ; dl = physical drive number (0,1, 80h, 81h, 82h, 83h)
920     <1> mov     ah, 18h ; LBA read
921     <1> mov     al, 1 ; sector count
922     <1> call    int13h
923     <1> jc     short loc_validate_hd_fs_partition_err_retn
924     <1> loc_hd_drv_fs_boot_validation:
925     <1> ;mov     esi, DOSBootSectorBuff
926     <1> mov     esi, ebx ; Boot sector buffer
927     <1> cmp     word [esi+BS_validation], 0AA55h
928     <1> jne     short loc_validate_hd_fs_partition_stc_retn
929     <1> ;
930     <1> ;Singlix FS Extensions to TR-DOS (7/6/2009)
931     <1> cmp     word [esi+bs_FS_Identifier], 'FS' ; 03/02/2018
932     <1> jne     short loc_validate_hd_fs_partition_stc_retn
933     <1> ; 'A1h' check is not necessary
934     <1> ; if 'FS' check is passed as OK/Yes.
935     <1> cmp     byte [esi+bs_FS_PartitionID], 0A1h
936     <1> jne     short loc_validate_hd_fs_partition_stc_retn
937     <1> ;
938     <1> mov     edi, ebp ; 10/01/2016
939     <1> ;
940     <1> mov     al, byte [esi+bs_FS_LBA_Ready]
941     <1> mov     [edi+LD_FS_LBAYes], al
942     <1> ;
943     <1> ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix
944     <1> mov     al, [esi+bs_FS_MediaAttrib]
945     <1> mov     byte [edi+LD_FS_MediaAttrib], al
946     <1> ;
947     <1> mov     al, [esi+bs_FS_VersionMaj]
948     <1> mov     [edi+LD_FS_VersionMajor], al
949     <1> ;
950     <1> mov     ax, [esi+bs_FS_BytesPerSec]
951     <1> mov     [edi+LD_FS_BytesPerSec], ax
952     <1> mov     al, [esi+bs_FS_SecPerTrack]
953     <1> xor     ah, ah ; 09/12/2017
954     <1> mov     [edi+LD_FS_SecPerTrack], ax
955     <1> mov     al, [esi+bs_FS_Heads]
956     <1> mov     [edi+LD_FS_NumHeads], ax
957     <1> ;
958     <1> mov     eax, [esi+bs_FS_UnDe1DirD]
959     <1> mov     [edi+LD_FS_UnDe1DirD], eax
960     <1> mov     edx, [esi+bs_FS_MATLocation]
961     <1> mov     [edi+LD_FS_MATLocation], edx
962     <1> mov     eax, [esi+bs_FS_RootDirD]
963     <1> mov     [edi+LD_FS_RootDirD], eax
964     <1> mov     eax, [esi+bs_FS_BeginSector]
965     <1> mov     [edi+LD_FS_BeginSector], eax
966     <1> mov     eax, [esi+bs_FS_VolumeSize]
967     <1> mov     [edi+LD_FS_VolumeSize], eax
968     <1> ;
969     <1> mov     eax, edx ; [edi+LD_FS_MATLocation]
970     <1> add     eax, [edi+LD_FS_BeginSector]
971     <1> mov     esi, edi
972     <1> mread_hd_fs_MAT_sector:
973     <1> ;mov     ebx, DOSBootSectorBuff
974     <1> ;mov     ecx, 1
975     <1> ; 25/07/2022
976     <1> sub     ecx, ecx
977     <1> inc     cl
978     <1> ; ecx = 1
979     <1> call    disk_read
980     <1> jc     short loc_validate_hd_fs_partition_retn
981     <1> ; EDI will not be changed
982     <1> mov     esi, ebx
983     <1> use_hdfs_mat_sector_params:
984     <1> mov     eax, [esi+FS_MAT_DATLocation]
985     <1> mov     [edi+LD_FS_DATLocation], eax
986     <1> mov     eax, [esi+FS_MAT_DATScout]
987     <1> mov     [edi+LD_FS_DATSectors], eax
988     <1> mov     eax, [esi+FS_MAT_FreeSectors]
989     <1> mov     [edi+LD_FS_FreeSectors], eax
990     <1> mov     eax, [esi+FS_MAT_FirstFreeSector]
991     <1> mov     [edi+LD_FS_FirstFreeSector], eax
992     <1> mov     eax, [esi+LD_FS_RootDirD]
993     <1> add     eax, [edi+LD_FS_BeginSector]
994     <1> mov     esi, edi
995     <1> read_hd_fs_RDT_sector:
996     <1> mov     ebx, DOSBootSectorBuff
997     <1> ;mov     ecx, 1
998     <1> mov     cl, 1
999     <1> call    disk_read
1000    <1> jc     short loc_validate_hd_fs_partition_retn
1001    <1> ; EDI will not be changed
1002    <1> mov     esi, ebx
1003    <1> use_hdfs_RDT_sector_params:
1004    <1> mov     eax, [esi+FS_RDT_VolumeSerialNo]
1005    <1> mov     [edi+LD_FS_VolumeSerial], eax
1006    <1> push    edi
1007    <1> ;mov     ecx, 16
1008    <1> mov     cl, 16
1009    <1> add     esi, FS_RDT_VolumeName
1010    <1> add     edi, LD_FS_VolumeName
1011    <1> rep     movsd ; 64 bytes
1012    <1> pop     esi
1013    <1> ; Volume Name Reset
1014    <1> mov     byte [esi+LD_FS_MediaChanged], 6
1015    <1> ;
1016    <1> ;mov     c1, [Last_DOS_DiskNo]
1017    <1> ;add     cl, 'A'
1018    <1> ;mov     [esi+LD_FS_Name], c1
1019    <1>
1020    <1> loc_validate_hd_fs_partition_retn:
1021    <1> retn
1022    <1>
1023    <1> %endif
1024    <1>
1025    <1> load_masterboot:
1026    <1> ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
1027    <1> ; 14/07/2020 (Reset function has been removed)
1028    <1> ;
1029    <1> ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1030    <1> ; 2005 - 2011
1031    <1> ; input -> DL = drive number
1032    <1> ; mov     ah, 0Dh ; Alternate disk reset
1033    <1> ; call    int13h
1034    <1> ; jnc     short pass_reset_error
1035    <1> ;harddisk_error:
1036    <1> ; retn

```

```

1037 <1> ;pass_reset_error:
1038 000073C1 BB[62790100] <1> mov ebx, MasterBootBuff
1039 <1> ;mov ax, 0201h
1040 <1> ;mov cx, 1
1041 <1> ; 25/07/2022
1042 <1> ;xor ecx, ecx
1043 <1> ;inc cl
1044 000073C6 B101 <1> mov cl, 1
1045 <1> ; ecx = 1
1046 000073C8 89C8 <1> mov eax, ecx ; ch = cylinder = 0
1047 000073CA B402 <1> mov ah, 2 ; chs read
1048 <1> ; eax = 0201h
1049 000073CC 30F6 <1> xor dh, dh ; head = 0
1050 000073CE E838DCFFFF <1> call int13h
1051 000073D3 720C <1> jc short haddisk_error
1052 <1> ;
1053 000073D5 66813D[607B0100]55- <1> cmp word [MBIDCode], 0AA55h
1054 000073DD AA <1> ;
1055 000073DE 7401 <1> je short load_masterboot_ok
1056 000073E0 F9 <1> stc
1057 <1> haddisk_error:
1058 000073E1 C3 <1> load_masterboot_ok:
1059 <1> retn
1060 <1> get_free_FAT_sectors:
1061 <1> ; 29/08/2023
1062 <1> ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
1063 <1> ; 21/12/2017
1064 <1> ; 29/02/2016
1065 <1> ; 13/02/2016
1066 <1> ; 04/02/2016
1067 <1> ; 07/01/2016 (TRDOS 386 = TRDOS v2.0)
1068 <1> ; 11/07/2010
1069 <1> ; 21/06/2009
1070 <1> ; INPUT: ESI = Logical DOS Drive Description Table address
1071 <1> ; OUTPUT: STC => Error
1072 <1> ; cf = 0 and EAX = Free FAT sectors
1073 <1> ; Also, related parameters and FAT buffer will be reset and updated
1074 <1>
1075 000073E2 31C0 <1> xor eax, eax
1076 <1> ;mov [esi+LD_FreeSectors], eax ; Reset
1077 <1>
1078 000073E4 807E0302 <1> cmp byte [esi+LD_FATType], 2
1079 000073E8 7653 <1> jna short loc_gfc_get_fat_free_clusters
1080 <1>
1081 <1> ; 29/02/2016
1082 000073EA 48 <1> dec eax ; 0FFFFFFFh
1083 000073EB 89463A <1> mov [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count (reset)
1084 000073EE 89463E <1> mov [esi+LD_BPB+BPB_Reserved+4], eax ; First Free Cluster (reset)
1085 000073F1 40 <1> inc eax ; 0
1086 <1> ;
1087 000073F2 668B4636 <1> mov ax, [esi+LD_BPB+BPB_FSInfo]
1088 000073F6 03466C <1> add eax, [esi+LD_StartSector]
1089 <1>
1090 000073F9 BB[727C0100] <1> mov ebx, DOSBootSectorBuff
1091 <1> ;mov ecx, 1
1092 <1> ; 25/07/2022
1093 000073FE 31C9 <1> xor ecx, ecx
1094 00007400 FEC1 <1> inc cl
1095 <1> ; ecx = 1
1096 00007402 E8DCA50000 <1> call disk_read
1097 00007407 7301 <1> jnc short loc_gfc_check_fsinfo_signs
1098 <1> retn_gfc_get_fsinfo_sec:
1099 00007409 C3 <1> retn
1100 <1>
1101 <1> loc_gfc_check_fsinfo_signs:
1102 0000740A BB[727C0100] <1> mov ebx, DOSBootSectorBuff ; 13/02/2016
1103 0000740F 813B52526141 <1> cmp dword [ebx], 41615252h
1104 00007415 7524 <1> jne short retn_gfc_get_fsinfo_stc
1105 <1> ;add ebx, 484
1106 <1> ;cmp dword [ebx], 61417272h
1107 00007417 81BBE4010000727241- <1> cmp dword [ebx+484], 61417272h
1108 00007420 61 <1>
1109 <1> jne short retn_gfc_get_fsinfo_stc
1110 <1> ;add ebx, 4
1111 00007423 8B83E8010000 <1> ;mov eax, [ebx]
1112 <1> mov eax, [ebx+488]
1113 <1> ; 29/02/2016
1114 00007429 89463A <1> mov [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count
1115 0000742C 8B93EC010000 <1> mov edx, [ebx+492]
1116 <1> ; 29/08/2023 (BugFix)
1117 00007432 89563E <1> mov [esi+LD_BPB+BPB_Reserved+4], edx ; First Free Cluster
1118 <1> ; 21/12/2017
1119 00007435 89C3 <1> mov ebx, eax ; (initial value = 0FFFFFFFh)
1120 00007437 43 <1> inc ebx ; 0FFFFFFFh -> 0
1121 00007438 7513 <1> jnz short short retn_from_get_free_fat32_clusters
1122 0000743A C3 <1> retn
1123 <1>
1124 0000743B F9 <1> retn_gfc_get_fsinfo_stc:
1125 0000743C C3 <1> stc
1126 <1> retn
1127 <1>
1128 <1> loc_gfc_get_fat_free_clusters:
1129 0000743D B002 <1> ;mov eax, 2
1130 <1> mov al, 2
1131 <1> ;mov [FAT_CurrentCluster], eax
1132 0000743F E835480000 <1> loc_gfc_loop_get_next_cluster:
1133 00007444 730E <1> call get_next_cluster
1134 00007446 21C0 <1> jnc short loc_gfc_free_fat_clusters_cont
1135 00007448 7411 <1> and eax, eax
1136 <1> jz short loc_gfc_pass_inc_free_cluster_count
1137 <1>
1138 0000744A 8B4674 <1> retn_from_get_free_fat_clusters:
1139 <1> mov eax, [esi+LD_FreeSectors] ; Free clusters !
1140 0000744D 0FB65E13 <1> retn_from_get_free_fat32_clusters:
1141 00007451 F7E3 <1> movzx ebx, byte [esi+LD_BPB+BPB_SecPerClust]
1142 <1> mul ebx
1143 <1> ;mov [esi+LD_FreeSectors], eax ; Free sectors
1144 00007453 C3 <1> retn_get_free_sectors_calc:
1145 <1> retn
1146 <1>
1147 00007454 09C0 <1> loc_gfc_free_fat_clusters_cont:
1148 00007456 7503 <1> or eax, eax
1149 00007458 FF4674 <1> jnz short loc_gfc_pass_inc_free_cluster_count
1150 <1> inc dword [esi+LD_FreeSectors] ; Free clusters !
1151 <1>
1152 <1> loc_gfc_pass_inc_free_cluster_count:
1153 0000745B 89C8 <1> ;mov eax, [FAT_CurrentCluster]
1154 0000745D 3B4678 <1> mov eax, ecx ; [FAT_CurrentCluster]
1155 00007460 77E8 <1> cmp eax, [esi+LD_Clusters]
1156 00007462 40 <1> ja short retn_from_get_free_fat_clusters
1157 <1> inc eax
1158 00007463 EBDA <1> ;mov [FAT_CurrentCluster], eax
1159 <1> jmp short loc_gfc_loop_get_next_cluster

```

```

1159 <1>
1160 <1> floppy_drv_init:
1161 <1> ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
1162 <1> ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
1163 <1> ; 09/12/2017
1164 <1> ; 06/07/2016
1165 <1> ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1166 <1> ; 24/07/2011
1167 <1> ; 04/07/2009
1168 <1> ; INPUT ->
1169 <1> ; DL = Drive number (0,1)
1170 <1> ; OUTPUT ->
1171 <1> ; BL = drive name
1172 <1> ; BH = drive number
1173 <1> ; ESI = Logical DOS drv description table
1174 <1> ; EAX = Volume serial number
1175 <1>
1176 00007465 BE[7C660000] <1> mov esi, fd0_type ; 10/01/2016
1177 0000746A BF00010900 <1> mov edi, Logical_DOSDisks
1178 0000746F 08D2 <1> or dl, dl
1179 00007471 7407 <1> jz short loc_drv_init_fd0_fd1
1180 00007473 81C700010000 <1> add edi, 100h
1181 00007479 46 <1> inc esi ; fd1_type ; 10/01/2016
1182 <1> loc_drv_init_fd0_fd1:
1183 0000747A C6477E00 <1> mov byte [edi+LD_MediaChanged], 0
1184 0000747E 803E01 <1> cmp byte [esi], 1 ; type (>0 if it is existing)
1185 <1> ; 4 = 1.44 MB, 80 track, 3 1/2"
1186 00007481 7221 <1> jb short read_fd_boot_sector_retn
1187 00007483 885702 <1> mov [edi+LD_PhyDrvNo], dl
1188 <1> read_fd_boot_sector:
1189 00007486 30F6 <1> xor dh, dh
1190 00007488 B904000000 <1> mov ecx, 4 ; Retry Count
1191 <1> read_fd_boot_sector_again:
1192 0000748D 51 <1> push ecx
1193 <1> ;mov cx, 1
1194 0000748E B101 <1> mov cl, 1
1195 00007490 66B80102 <1> mov ax, 0201h ; Read 1 sector
1196 00007494 8B[727C0100] <1> mov ebx, DOSBootSectorBuff
1197 00007499 E86DDBFFFF <1> call int13h
1198 0000749E 59 <1> pop ecx
1199 0000749F 7304 <1> jnc short use_fd_boot_sector_params
1200 000074A1 E2EA <1> loop read_fd_boot_sector_again
1201 <1>
1202 <1> read_fd_boot_sector_stc_retn:
1203 000074A3 F9 <1> stc
1204 <1> read_fd_boot_sector_retn:
1205 000074A4 C3 <1> retn
1206 <1>
1207 <1> use_fd_boot_sector_params:
1208 <1> ;mov esi, DOSBootSectorBuff
1209 000074A5 89DE <1> mov esi, ebx
1210 000074A7 6681BEFE01000055AA <1> cmp word [esi+BS_Validation], 0AA55h
1211 000074B0 75F1 <1> jne short read_fd_boot_sector_stc_retn
1212 <1>
1213 <1> ; 19/12/2025
1214 <1> %if 0
1215 <1> cmp word [esi+bs_FS_Identifier], 'SF'
1216 <1> ;jne use_fd_fatfs_boot_sector_params
1217 <1> ; 25/07/2022
1218 <1> je short use_fdfs_boot_sector_params
1219 <1> jmp use_fd_fatfs_boot_sector_params
1220 <1> use_fdfs_boot_sector_params:
1221 <1> mov al, [esi+bs_FS_LBA_Ready]
1222 <1> mov [edi+LD_FS_LBAYes], al
1223 <1> ;
1224 <1> ; 03/01/2010 CHS -> DOS FAT/BPB compatibility fix
1225 <1> mov al, [esi+bs_FS_MediaAttrib]
1226 <1> mov [edi+LD_FS_MediaAttrib], al
1227 <1> ;
1228 <1> mov al, [esi+bs_FS_VersionMaj]
1229 <1> mov byte [edi+LD_FS_VersionMajor], al
1230 <1> mov ax, [esi+bs_FS_BytesPerSec]
1231 <1> mov [edi+LD_FS_BytesPerSec], ax
1232 <1> mov al, [esi+bs_FS_SecPerTrack]
1233 <1> sub ah, ah ; 09/12/2017
1234 <1> mov [edi+LD_FS_SecPerTrack], ax
1235 <1> mov al, [esi+bs_FS_Heads]
1236 <1> mov [edi+LD_FS_NumHeads], ax
1237 <1> ;
1238 <1> mov eax, [esi+bs_FS_UnDeDirD]
1239 <1> mov [edi+LD_FS_UnDeDirD], eax
1240 <1> mov eax, [esi+bs_FS_MATLocation]
1241 <1> mov [edi+LD_FS_MATLocation], eax
1242 <1> mov eax, [esi+bs_FS_RootDirD]
1243 <1> mov [edi+LD_FS_RootDirD], eax
1244 <1> mov eax, [esi+bs_FS_BeginSector]
1245 <1> mov [edi+LD_FS_BeginSector], eax
1246 <1> mov eax, [esi+bs_FS_VolumeSize]
1247 <1> mov [edi+LD_FS_VolumeSize], eax
1248 <1> ;
1249 <1> mov esi, edi
1250 <1> mov eax, [esi+LD_FS_MATLocation]
1251 <1> ;add eax, [edi+LD_FS_BeginSector]
1252 <1> read_fd_MAT_sector_again:
1253 <1> ;mov ebx, DOSBootSectorBuff
1254 <1> ;mov ecx, 1
1255 <1> mov cl, 1
1256 <1> call chs_read
1257 <1> mov esi, ebx
1258 <1> ;jnc short use_fdfs_mat_sector_params
1259 <1> ;jmp short read_fd_boot_sector_retn
1260 <1> ;retn
1261 <1> ; 25/07/2022
1262 <1> jc short read_fd_RDT_sector_retn
1263 <1> use_fdfs_mat_sector_params:
1264 <1> mov eax, [esi+FS_MAT_DATLocation]
1265 <1> mov [edi+LD_FS_DATLocation], eax
1266 <1> mov eax, [esi+FS_MAT_DATScout]
1267 <1> mov [edi+LD_FS_DATSectors], eax
1268 <1> mov eax, [esi+FS_MAT_FreeSectors]
1269 <1> mov [edi+LD_FS_FreeSectors], eax
1270 <1> mov eax, [esi+FS_MAT_FirstFreeSector]
1271 <1> mov [edi+LD_FS_FirstFreeSector], eax
1272 <1> ;
1273 <1> mov esi, edi
1274 <1> mov eax, [esi+LD_FS_RootDirD]
1275 <1> read_fd_RDT_sector_again:
1276 <1> ;mov ebx, DOSBootSectorBuff
1277 <1> ;mov cx, 1
1278 <1> mov cl, 1
1279 <1> call chs_read
1280 <1> mov esi, ebx
1281 <1> jc short read_fd_RDT_sector_retn
1282 <1> use_fdfs_RDT_sector_params:

```

```

1283     <1>     mov     eax, [esi+FS_RDT_VolumeSerialNo]
1284     <1>     mov     [edi+LD_FS_VolumeSerial], eax
1285     <1>     push    edi
1286     <1>     ;mov    ecx, 16
1287     <1>     mov     cl, 16
1288     <1>     add     esi, FS_RDT_VolumeName
1289     <1>     add     edi, LD_FS_VolumeName
1290     <1>     rep     movsd ; 64 bytes
1291     <1>     pop     esi
1292     <1>     mov     byte [esi+LD_FATType], 0
1293     <1>     mov     byte [esi+LD_FSType], 0A1h
1294     <1>     jmp     loc_cont_use_fd_boot_sector_params
1295     <1>
1296     <1> read_fd_RDT_sector_stc_retn:
1297     <1>     stc
1298     <1> read_fd_RDT_sector_retn:
1299     <1>     retn
1300     <1> %endif
1301     <1>
1302     <1> use_fd_fatfs_boot_sector_params:
1303     <1>     cmp     byte [esi+BS_BootSig], 29h
1304     <1>     ;jne     short read_fd_RDT_sector_stc_retn
1305     <1>     ; 19/12/2025
1306     <1>     jne     short read_fd_boot_sector_stc_retn
1307     <1>     cmp     byte [esi+BPB_Media], 0F0h
1308     <1>     ;jb     short read_fd_RDT_sector_retn
1309     <1>     ; 19/12/2025
1310     <1>     jb     short read_fd_boot_sector_retn
1311     <1>     push    edi
1312     <1>     add     edi, LD_BPB
1313     <1>     ;mov    ecx, 16
1314     <1>     mov     cl, 16
1315     <1>     rep     movsd ; 64 bytes
1316     <1>     pop     esi
1317     <1>     xor     eax, eax
1318     <1>     mov     [esi+LD_StartSector], eax ; 0
1319     <1>     mov     ax, [esi+LD_BPB+BPB_FATSz16]
1320     <1>     mov     cl, [esi+LD_BPB+BPB_NumFATS]
1321     <1>     mul     ecx
1322     <1>     ; edx = 0 !
1323     <1>     mov     dx, [esi+LD_BPB+BPB_RsvdSecCnt]
1324     <1>     mov     [esi+LD_FATBegin], dx
1325     <1>     ; 25/07/2022
1326     <1>     add     eax, edx
1327     <1>     ;add    ax, dx
1328     <1>     mov     [esi+LD_ROOTBegin], eax
1329     <1>     mov     [esi+LD_DATABegin], eax
1330     <1>     mov     dx, [esi+LD_BPB+BPB_RootEntCnt]
1331     <1>     ;shl    edx, 5 ; * 32 (Size of a directory entry)
1332     <1>     ;shl    dx, 5
1333     <1>     ;add    edx, 511
1334     <1>     ;add    dx, 511
1335     <1>     ;shr    edx, 9 ; edx = ((edx*32)+511) / 512
1336     <1>     ;shr    dx, 9
1337     <1>     add     dx, 15 ; 06/07/2016 (+512/32)-1
1338     <1>     ;shr    dx, 4 ; / 16 (==16 entries per sector)
1339     <1>     ; 25/07/2022
1340     <1>     shr     edx, 4
1341     <1>     add     [esi+LD_DATABegin], edx ; + rd sectors
1342     <1>     ;movzx   eax, word [esi+LD_BPB+BPB_TotalSec16]
1343     <1>     mov     ax, [esi+LD_BPB+BPB_TotalSec16]
1344     <1>     mov     [esi+LD_TotalSectors], eax
1345     <1>     sub     eax, [esi+LD_DATABegin]
1346     <1>     ;movzx   ecx, byte [esi+LD_BPB+BPB_SecPerClust]
1347     <1>     mov     cl, [esi+LD_BPB+BPB_SecPerClust]
1348     <1>     cmp     cl, 1
1349     <1>     jna     short save_fd_fatfs_cluster_count
1350     <1>     ; 25/07/2022
1351     <1>     sub     edx, edx
1352     <1>     ;sub    dx, dx ; 0
1353     <1>     ;sub    dl, dl ; 06/07/2016
1354     <1>     div     ecx
1355     <1> save_fd_fatfs_cluster_count:
1356     <1>     mov     [esi+LD_Clusters], eax
1357     <1>
1358     <1>     ; Maximum valid cluster Number = EAX +1
1359     <1>     ; with 2 reserved clusters = EAX +2
1360     <1>
1361     <1> reset_FAT_buffer_decriptors:
1362     <1>     sub     eax, eax ; 0
1363     <1>     mov     [FAT_BuffValidData], al ; 0
1364     <1>     mov     [FAT_BuffDrvName], al ; 0
1365     <1>     mov     [FAT_BuffSector], eax ; 0
1366     <1>
1367     <1> read_fd_FAT_sectors:
1368     <1>     mov     ebx, FAT_Buffer
1369     <1>     mov     ax, [esi+LD_BPB+BPB_RsvdSecCnt]
1370     <1>     ;mov    ecx, 3
1371     <1>     mov     cl, 3 ; 3 sectors
1372     <1>     call    chs_read
1373     <1>     jc     short read_fd_FAT_sectors_retn
1374     <1> use_fd_FAT_sectors:
1375     <1>     mov     al, [esi+LD_PhyDrvNo]
1376     <1>     add     al, 'A'
1377     <1>     mov     [FAT_BuffDrvName], al
1378     <1>     mov     byte [FAT_BuffValidData], 1
1379     <1>     call    fd_init_calculate_free_clusters
1380     <1>     jc     short read_fd_FAT_sectors_retn
1381     <1>
1382     <1> loc_use_fd_boot_sector_params_FAT:
1383     <1>     mov     byte [esi+LD_FATType], 1 ; FAT 12
1384     <1>     mov     byte [esi+LD_FSType], 1
1385     <1>     mov     eax, [esi+LD_BPB+VolumeID]
1386     <1> loc_cont_use_fd_boot_sector_params:
1387     <1>     mov     bh, [esi+LD_PhyDrvNo]
1388     <1>     mov     [esi+LD_DParamEntry], bh
1389     <1>     mov     bl, bh
1390     <1>     add     bl, 'A'
1391     <1>     mov     byte [esi+LD_Name], bl
1392     <1>     mov     byte [esi+LD_DiskType], 1
1393     <1>     mov     byte [esi+LD_LBAYes], 0
1394     <1>     mov     byte [esi+LD_PartitionEntry], 0
1395     <1>     mov     byte [esi+LD_MediaChanged], 6 ; Volume Name Reset
1396     <1>
1397     <1> read_fd_FAT_sectors_retn:
1398     <1>     retn
1399     <1>
1400     <1> fd_init_calculate_free_clusters:
1401     <1>     ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
1402     <1>     ; 09/12/2017
1403     <1>     ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1404     <1>     ; 04/07/2009
1405     <1>     ; INPUT ->
1406     <1>     ; ESI = Logical DOS drive description table address

```



```

1407      <1>      ; OUTPUT ->
1408      <1>      ;      [ESI+LD_FreeSectors] will be set
1409      <1>
1410      <1>      sub     eax, eax
1411      <1>      mov     [esi+LD_FreeSectors], eax ; 0
1412      <1>      mov     al, 2 ; eax = 2
1413      <1>
1414      <1>      fd_init_loop_get_next_cluster:
1415      <1>      call    fd_init_get_next_cluster
1416      <1>      jc      short fd_init_calculate_free_clusters_retn
1417      <1>
1418      <1>      fd_init_free_fat_clusters:
1419      <1>      ;cmp     eax, 0
1420      <1>      ;ja      short fd_init_pass_inc_free_cluster_count
1421      <1>      ;and     eax, eax
1422      <1>      ;jnz     short fd_init_pass_inc_free_cluster_count
1423      <1>      ;and     ax, ax
1424      <1>      and     eax, eax ; 25/07/2022
1425      <1>      jnz     short fd_init_pass_inc_free_cluster_count
1426      <1>      ; 25/07/2022
1427      <1>      inc     dword [esi+LD_FreeSectors]
1428      <1>      ;inc     word [esi+LD_FreeSectors]
1429      <1>
1430      <1>      fd_init_pass_inc_free_cluster_count:
1431      <1>      ; 25/07/2022
1432      <1>      mov     eax, [FAT_CurrentCluster]
1433      <1>      ;mov     ax, [FAT_CurrentCluster]
1434      <1>      cmp     eax, [esi+LD_Clusters]
1435      <1>      ;cmp     ax, [esi+LD_Clusters]
1436      <1>      ja      short retn_from_fd_init_calculate_free_clusters
1437      <1>      inc     eax
1438      <1>      ;inc     ax
1439      <1>      jmp     short fd_init_loop_get_next_cluster
1440      <1>
1441      <1>      retn_from_fd_init_calculate_free_clusters:
1442      <1>      ; 25/07/2022
1443      <1>      ;xor     eax, eax
1444      <1>      xor     ah, ah
1445      <1>      mov     al, [esi+LD_BPB+BPB_SecPerClust]
1446      <1>      cmp     al, 1
1447      <1>      jna      short fd_init_calculate_free_clusters_retn
1448      <1>      ;movzx   eax, al
1449      <1>      ;xor     ah, ah ; 09/12/2017
1450      <1>      ; 25/07/2022
1451      <1>      mov     ecx, [esi+LD_FreeSectors]
1452      <1>      ;mov     cx, [esi+LD_FreeSectors] ; Count of free clusters
1453      <1>      mul     ecx
1454      <1>      ;mul     cx
1455      <1>      mov     [esi+LD_FreeSectors], eax
1456      <1>      ;mov     [esi+LD_FreeSectors], ax
1457      <1>      fd_init_calculate_free_clusters_retn:
1458      <1>      retn
1459      <1>
1460      <1>      fd_init_get_next_cluster:
1461      <1>      ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
1462      <1>      ; 04/02/2016
1463      <1>      ; 02/02/2016
1464      <1>      ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1465      <1>      ; 04/07/2009
1466      <1>      ; INPUT ->
1467      <1>      ;      EAX = Current cluster
1468      <1>      ;      ESI = Logical DOS drive description table address
1469      <1>      ;      EDX = 0
1470      <1>      ; OUTPUT ->
1471      <1>      ;      EAX = Next cluster
1472      <1>
1473      <1>      mov     [FAT_CurrentCluster], eax
1474      <1>      fd_init_get_next_cluster_readnext:
1475      <1>      sub     edx, edx ; 0
1476      <1>      mov     ebx, 1024 ; 400h
1477      <1>      div     ebx
1478      <1>      ; EAX = Count of 3 FAT sectors
1479      <1>      ; EDX = Buffer entry index
1480      <1>      mov     ecx, eax
1481      <1>      ;mov     eax, 3
1482      <1>      mov     al, 3
1483      <1>      mul     edx ; Multiply by 3
1484      <1>      ;shr     ax, 1 ; Divide by 2
1485      <1>      ; 25/07/2022
1486      <1>      shr     eax, 1
1487      <1>      mov     ebx, eax ; Buffer byte offset
1488      <1>      add     ebx, FAT_Buffer
1489      <1>      mov     eax, ecx
1490      <1>      ;mov     edx, 3
1491      <1>      mov     dx, 3
1492      <1>      mul     edx
1493      <1>      ; EAX = FAT Beginning Sector
1494      <1>      ; EDX = 0
1495      <1>      mov     cl, [esi+LD_Name]
1496      <1>      ;cmp     byte [FAT_BuffValidData], 0
1497      <1>      ;jna      short fd_init_load_FAT_sectors0
1498      <1>      cmp     cl, [FAT_BuffDrvName]
1499      <1>      ;jne      short fd_init_load_FAT_sectors0
1500      <1>      cmp     eax, [FAT_BuffSector]
1501      <1>      ;jne      short fd_init_load_FAT_sectors1
1502      <1>      ; 25/07/2022
1503      <1>      mov     eax, [FAT_CurrentCluster]
1504      <1>      ;mov     al, [FAT_CurrentCluster]
1505      <1>      shr     eax, 1
1506      <1>      ;shr     al, 1
1507      <1>      mov     ax, [ebx]
1508      <1>      jnc     short fd_init_gnc_even
1509      <1>      ;shr     ax, 4
1510      <1>      ; 25/04/2022
1511      <1>      shr     eax, 4
1512      <1>      fd_init_gnc_clc_retn:
1513      <1>      clc
1514      <1>      retn
1515      <1>
1516      <1>      fd_init_gnc_even:
1517      <1>      and     ah, 0Fh
1518      <1>      retn
1519      <1>
1520      <1>      fd_init_load_FAT_sectors0:
1521      <1>      mov     [FAT_BuffDrvName], cl
1522      <1>      fd_init_load_FAT_sectors1:
1523      <1>      mov     byte [FAT_BuffValidData], 0
1524      <1>      mov     [FAT_BuffSector], eax
1525      <1>      add     eax, [esi+LD_FATBegin]
1526      <1>      mov     ebx, FAT_Buffer
1527      <1>      ;movzx   ecx, word [esi+LD_BPB+BPB_FATSz16]
1528      <1>      mov     cx, [esi+LD_BPB+BPB_FATSz16]
1529      <1>      ;sub     cx, [FAT_BuffSector]
1530      <1>      ; 25/07/2022

```

```

1531 0000760F 2B0D[7A7E0100]    <1>    sub     ecx, [FAT_BuffSector]
1532                                <1>    ;sub     edx, edx
1533 00007615 B203              <1>    mov     dl, 3
1534                                <1>    ; edx = 3
1535                                <1>    ;;cmp    ecx, 3
1536                                <1>    ;cmp     cx, 3
1537 00007617 39D1              <1>    cmp     ecx, edx ; 3
1538 00007619 7602              <1>    jna     short fdinit_pass_fix_sector_count_3
1539                                <1>    ;;mov    ecx, 3
1540                                <1>    ;mov     ecx, 3
1541 0000761B 89D1              <1>    mov     ecx, edx ; 3
1542                                <1>    fdinit_pass_fix_sector_count_3:
1543 0000761D E8C7A30000          <1>    call    chs_read
1544 00007622 730D              <1>    jnc     short fd_init_FAT_sectors_no_load_error
1545 00007624 C605[767E0100]00    <1>    mov     byte [FAT_BuffValidData], 0
1546                                <1>    ; Drv not ready or read Error !
1547 0000762B B80F000000          <1>    mov     eax, ERR_DRV_NOT_RDY ; 15
1548                                <1>    ;xor     edx, edx
1549 00007630 C3                <1>    retn
1550                                <1>
1551                                <1>    fd_init_FAT_sectors_no_load_error:
1552 00007631 C605[767E0100]01    <1>    mov     byte [FAT_BuffValidData], 1
1553 00007638 A1[727E0100]        <1>    mov     eax, [FAT_CurrentCluster]
1554 0000763D E967FFFFFF          <1>    jmp     fd_init_get_next_cluster_readnext
1555                                <1>
1556                                <1>    get_FAT_volume_name:
1557                                <1>    ; 19/12/2025 (TRDOS 386 Kernel v2.0.10)
1558                                <1>    ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
1559                                <1>    ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1560                                <1>    ; 12/09/2009
1561                                <1>    ; INPUT ->
1562                                <1>    ;     BH = Logical DOS drive number (0,1,2,3,4 ...)
1563                                <1>    ;     BL = 0
1564                                <1>    ; OUTPUT ->
1565                                <1>    ;     CF = 0 -> ESI = Volume name address
1566                                <1>    ;     CF = 1 -> Root volume name not found
1567                                <1>
1568                                <1>    ;mov     ah, 0FFh
1569                                <1>    ;mov     al, [Last_Dos_DiskNo]
1570                                <1>    ;cmp     al, bh
1571                                <1>    ;jb     short loc_gfvn_dir_load_err
1572                                <1>
1573 00007642 89DE              <1>    mov     esi, ebx
1574 00007644 81E600FF0000        <1>    and     esi, 0FF00h ; esi = bh
1575 0000764A 81C600010900        <1>    add     esi, Logical_DOSDisks
1576 00007650 8A06              <1>    mov     al, [esi+LD_Name]
1577 00007652 8A6603              <1>    mov     ah, [esi+LD_FATType]
1578                                <1>    ; 19/12/2025
1579                                <1>    ;cmp     ah, 1
1580                                <1>    ;jb     short loc_gfvn_dir_load_err
1581 00007655 3C41              <1>    cmp     al, 'A'
1582 00007657 720C              <1>    jb     short loc_gfvn_dir_load_err
1583 00007659 80FC02              <1>    cmp     ah, 2
1584 0000765C 7708              <1>    ja     short get_FAT32_root_cluster
1585                                <1>
1586 0000765E E8544A0000          <1>    call    load_FAT_root_directory
1587 00007663 730B              <1>    jnc     short loc_get_volume_name
1588                                <1>
1589                                <1>    loc_gfvn_dir_load_err:
1590                                <1>    loc_get_volume_name_retn: ; 29/08/2023
1591 00007665 C3                <1>    retn
1592                                <1>
1593                                <1>    get_FAT32_root_cluster:
1594 00007666 8B4632              <1>    mov     eax, [esi+LD_BPB+BPB_RootClus]
1595 00007669 E8C74A0000          <1>    call    load_FAT_sub_directory
1596 0000766E 72F5              <1>    jc     short loc_get_volume_name_retn
1597                                <1>
1598                                <1>    loc_get_volume_name:
1599                                <1>    mov     esi, Directory_Buffer
1600                                <1>    ;xor     cx, cx ; 0
1601                                <1>    ; 25/07/2022
1602 00007675 31C9              <1>    xor     ecx, ecx ; 0
1603                                <1>    check_root_volume_name:
1604 00007677 8A06              <1>    mov     al, [esi]
1605 00007679 08C0              <1>    or      al, al
1606 0000767B 74E8              <1>    jz     short loc_get_volume_name_retn
1607 0000767D 807E0B08          <1>    cmp     byte [esi+0Bh], 08h
1608 00007681 74E2              <1>    je     short loc_get_volume_name_retn
1609 00007683 663B0D[8C7E0100]    <1>    cmp     cx, [DirBuff_LastEntry]
1610 0000768A 7306              <1>    jnb     short pass_check_root_volume_name
1611                                <1>    ;inc     cx
1612                                <1>    ; 25/07/2022
1613 0000768C 41                <1>    inc     ecx
1614 0000768D 83C620          <1>    add     esi, 32
1615 00007690 EBE5              <1>    jmp     short check_root_volume_name
1616                                <1>
1617                                <1>    ; 29/08/2023
1618                                <1>    ;loc_get_volume_name_retn:
1619                                <1>    ;retn
1620                                <1>
1621                                <1>    pass_check_root_volume_name:
1622 00007692 803D[887E0100]03    <1>    cmp     byte [DirBuff_FATType], 3
1623 00007699 7230              <1>    jb     short loc_get_volume_name_retn_xor
1624                                <1>
1625 0000769B BB001C0900          <1>    mov     ebx, FAT_Buffer
1626 000076A0 BE00010900          <1>    mov     esi, Logical_DOSDisks
1627 000076A5 31C0              <1>    xor     eax, eax
1628 000076A7 8A25[877E0100]    <1>    mov     ah, [DirBuff_DRV]
1629 000076AD 80EC41              <1>    sub     ah, 'A'
1630 000076B0 01C6              <1>    add     esi, eax
1631 000076B2 A1[8E7E0100]        <1>    mov     eax, [DirBuff_Cluster]
1632 000076B7 E8BD480000          <1>    call    get_next_cluster
1633 000076BC 7305              <1>    jnc     short loc_gfvn_load_FAT32_dir_cluster
1634                                <1>
1635 000076BE 83F801              <1>    cmp     eax, 1
1636 000076C1 F5                <1>    cmc
1637 000076C2 C3                <1>    retn
1638                                <1>
1639                                <1>    loc_gfvn_load_FAT32_dir_cluster:
1640 000076C3 E86D4A0000          <1>    call    load_FAT_sub_directory
1641 000076C8 73A6              <1>    jnc     short loc_get_volume_name
1642 000076CA C3                <1>    retn
1643                                <1>
1644                                <1>    loc_get_volume_name_retn_xor:
1645 000076CB 31C0              <1>    xor     eax, eax
1646 000076CD C3                <1>    retn
1647                                <1>
1648                                <1>    get_media_change_status:
1649                                <1>    ; 10/01/2016 (TRDOS 386 = TRDOS v2.0)
1650                                <1>    ; 09/09/2009
1651                                <1>    ; INPUT:
1652                                <1>    ;     DL = Drive number (physical)
1653                                <1>    ; OUTPUT: c1c & AH = 6 media changed
1654                                <1>    ;     c1c & AH = 0 media not changed

```

```

1655      ;      stc -> Drive not ready or an error
1656
1657      mov     ah, 16h
1658      call    int13h
1659      cmp     ah, 06h
1660      je      short loc_gmc_status_retn
1661      or      ah, ah
1662      jz      short loc_gmc_status_retn
1663      loc_gmc_status_stc_retn:
1664      stc
1665      loc_gmc_status_retn:
1666      retn
3434      %include 'trdosk3.s' ; 06/01/2016
1      ; *****
2      ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.10) - MAIN PROGRAM : trdosk3.s
3      ; -----
4      ; Last Update: 19/12/2025 (Previous: 26/09/2024, TRDOS 386 v2.0.9)
5      ; -----
6      ; Beginning: 06/01/2016
7      ; -----
8      ; Assembler: NASM version 2.15 (trdos386.s)
9      ; -----
10     ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
11     ; MAINPROG.ASM (09/11/2011)
12     ; *****
13     ; MAINPROG.ASM [ TRDOS KERNEL - COMMAND EXECUTER SECTION - MAIN PROGRAM ]
14     ; (C) 2004-2011 Erdogan TAN [ 17/01/2004 ] Last Update: 09/11/2011
15     ; CMD_INTR.ASM [ TRDOS Command Interpreter Procedure ] Last Update: 09/11/2011
16     ; DIR.ASM [ DIRECTORY FUNCTIONS ] Last Update: 09/10/2011
17     ; FILE.ASM [ FILE FUNCTIONS ] Last Update: 09/10/2011
18
19     ; change_current_drive:
20     ; 19/12/2025 (TRDOS 386 Kernel v2.0.10)
21     ; 26/07/2022 (TRDOS 386 Kernel v2.0.5)
22     ; 16/10/2016
23     ; 02/02/2016
24     ; 15/01/2016 (TRDOS 386 = TRDOS v2.0)
25     ; 18/08/2011
26     ; 09/09/2009
27     ; INPUT:
28     ; DL = Logical DOS Drive Number
29     ; OUTPUT:
30     ; cf=1 -> Not successful
31     ; EAX = Error code
32     ; cf=0 ->
33     ; EAX = 0 (successful)
34
35     xor     ebx, ebx
36     mov     bh, dl
37
38     ; cmp     dl, 1
39     ; jna     short loc_ccdrv_initial_media_change_check
40     ; cmp     bh, [Last_Dos_DiskNo]
41     ; ja      short loc_ccdrv_drive_not_ready_err
42
43     loc_ccdrv_initial_media_change_check:
44     mov     esi, Logical_DOSDisks
45     add     esi, ebx
46     loc_ccdrv_dos_drive_name_check:
47     cmp     dl, 2
48     jb      short loc_ccdrv_dos_drive_name_check_ok
49
50     mov     al, [esi+LD_Name]
51     sub     al, 'A'
52     cmp     al, dl
53     je      short loc_ccdrv_dos_drive_name_check_ok
54
55     loc_ccdrv_drive_not_ready_err:
56     ; 16/10/2016 (15h -> 15)
57     ; mov     eax, 15 ; Drive not ready
58     ; 26/07/2022
59     sub     eax, eax
60     mov     al, 15
61     loc_change_current_drive_stc_retn:
62     stc
63     retn
64
65     loc_ccdrv_dos_drive_name_check_ok:
66     mov     ah, [esi+LD_MediaChanged]
67     cmp     ah, 6 ; VOLUME NAME CHECK/MOVE SIGN
68     je      short loc_ccdrv_get_FAT_volume_name_0
69
70     cmp     dl, 1
71     ja      short loc_gmcs_init_drv_hd
72
73     loc_gmcs_init_drv_fd:
74     or      ah, ah
75     ; AH = 1 is initialization sign (invalid_fd_parameter)
76     jnz     short loc_ccdrv_call_fd_init
77
78     call     get_media_change_status
79     jc      short loc_ccdrv_drive_not_ready_err
80
81     and     ah, ah
82     jz      short loc_change_current_drv3
83
84     xor     ah, 6
85     jnz     short loc_ccdrv_drive_not_ready_err
86
87     loc_ccdrv_call_fd_init_check_vol_id:
88     call     get_volume_serial_number
89     jnc     short loc_ccdrv_check_vol_serial
90
91     loc_ccdrv_call_fd_init:
92     call     floppy_drv_init
93     jnc     short loc_reset_drv_fd_current_dir
94
95     loc_ccdrv_fdinit_fail_retn:
96     ; 16/10/2016
97     mov     eax, 15 ; Drive not ready
98     retn
99
100    loc_ccdrv_check_vol_serial:
101    mov     [Current_VolSerial], eax
102    ; mov     dl, bh
103    call     floppy_drv_init
104    jc      short loc_ccdrv_fdinit_fail_retn
105
106    cmp     eax, [Current_VolSerial]
107    je      short loc_change_current_drv2
108
109    loc_reset_drv_fd_current_dir:
110    xor     eax, eax
111    mov     [esi+LD_CDirLevel], al

```

```

112 0000774C 89F7          <1>    mov     edi, esi
113 0000774E 81C780000000          <1>    add     edi, LD_CurrentDirectory
114 00007754 B920000000          <1>    mov     ecx, 32
115 00007759 F3AB          <1>    rep     stosd
116          <1>
117          <1> loc_ccdrv_get_FAT_volume_name_0:
118          <1> ; 19/12/2025
119          <1> ;mov     al, [esi+LD_FATType]
120          <1> ;or      al, al
121          <1> ;jz      short loc_change_current_drv2
122          <1>
123 0000775B 56          <1>    push    esi
124          <1> ;cmp     al, 2
125          <1> ; 19/12/2025
126 0000775C 807E0302          <1>    cmp     byte [esi+LD_FATType], 2
127 00007760 7705          <1>    ja      short loc_ccdrv_get_FAT32_vol_name
128          <1>
129          <1> loc_ccdrv_get_FAT2_16_vol_name:
130          <1> add     esi, LD_BPB + VolumeLabel
131 00007765 EB03          <1>    jmp     short loc_ccdrv_get_FAT_volume_name_1
132          <1>
133          <1> loc_ccdrv_get_FAT32_vol_name:
134 00007767 83C64D          <1>    add     esi, LD_BPB + FAT32_VolLab
135          <1> loc_ccdrv_get_FAT_volume_name_1:
136          <1> push    ebx
137 0000776B 56          <1>    push    esi
138 0000776C E8D1FEFFFF          <1>    call    get_FAT_volume_name
139 00007771 5F          <1>    pop     edi
140 00007772 5B          <1>    pop     ebx
141          <1> ; BL = 0
142 00007773 720B          <1>    jc      short loc_change_current_drv1
143 00007775 20C0          <1>    and     al, al
144 00007777 7407          <1>    jz      short loc_change_current_drv1
145          <1>
146          <1> loc_ccdrv_move_FAT_volume_name:
147 00007779 B90B000000          <1>    mov     ecx, 11
148 0000777E F3A4          <1>    rep     movsb
149          <1>
150          <1> loc_change_current_drv1:
151 00007780 5E          <1>    pop     esi
152 00007781 EB04          <1>    jmp     short loc_change_current_drv2
153          <1>
154          <1> loc_gmcs_init_drv_hd:
155 00007783 08E4          <1>    or      ah, ah
156 00007785 7404          <1>    jz      short loc_change_current_drv3
157          <1> ; BL = 0, BH = Logical DOS drive number
158          <1> loc_change_current_drv2:
159 00007787 C6467E00          <1>    mov     byte [esi+LD_MediaChanged], 0
160          <1> loc_change_current_drv3:
161 0000778B 883D[62770100]          <1>    mov     [Current_Drv], bh
162          <1>
163          <1> ;call    restore_current_directory
164          <1> ;retn
165          <1>
166          <1> restore_current_directory:
167          <1> ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
168          <1> ; 26/07/2022 (TRDOS 386 kernel v2.0.5)
169          <1> ; 11/02/2016
170          <1> ; 15/01/2016 (TRDOS 386 = TRDOS v2.0)
171          <1> ; 25/01/2010
172          <1> ; 12/10/2009
173          <1> ;
174          <1> ; INPUT:
175          <1> ; ESI = Logical DOS Drive Description Table
176          <1> ;
177          <1> ; OUTPUT:
178          <1> ; ESI = Logical DOS Drive Description Table
179          <1> ; EDI = offset Current_Dir_Drv
180          <1>
181 00007791 8A4603          <1>    mov     al, [esi+LD_FATType]
182 00007794 A2[61770100]          <1>    mov     [Current_FATType], al
183          <1>
184 00007799 8A26          <1>    mov     ah, [esi+LD_Name]
185 0000779B 8825[63770100]          <1>    mov     [Current_Dir_Drv], ah
186          <1>
187          <1> ; 19/12/2025
188          <1> ;and     al, al
189          <1> ;jz      short loc_restore_FS_current_directory
190          <1>
191          <1> loc_restore_FAT_current_directory:
192 000077A1 8A667F          <1>    mov     ah, [esi+LD_CDirLevel]
193 000077A4 8825[60770100]          <1>    mov     [Current_Dir_Level], ah
194 000077AA 08E4          <1>    or      ah, ah
195 000077AC 7410          <1>    jz      short loc_ccdrv_reset_cdir_FAT_12_16_32_fcluster
196          <1>
197 000077AE 0FB6D4          <1>    movzx   edx, ah
198 000077B1 C0E204          <1>    shl     dl, 4 ; * 16
199 000077B4 01F2          <1>    add     edx, esi
200 000077B6 8B828C000000          <1>    mov     eax, [edx+LD_CurrentDirectory+12]
201 000077BC EB24          <1>    jmp     short loc_ccdrv_reset_cdir_FAT_fcluster
202          <1>
203          <1> ; 19/12/2025
204          <1> %if 0
205          <1> loc_restore_FS_current_directory:
206          <1> ;call    load_current_FS_directory
207          <1> ;retn
208          <1> ; 26/07/2022
209          <1> jmp     load_current_FS_directory
210          <1> %endif
211          <1>
212          <1> loc_ccdrv_reset_cdir_FAT_12_16_32_fcluster:
213 000077BE 3C03          <1>    cmp     al, 3
214 000077C0 7205          <1>    jb      short loc_ccdrv_reset_cdir_FAT_12_16_fcluster
215          <1> loc_ccdrv_reset_cdir_FAT32_fcluster:
216 000077C2 8B4632          <1>    mov     eax, [esi+LD_BPB+FAT32_RootFClust]
217 000077C5 EB02          <1>    jmp     short loc_ccdrv_check_rootdir_sign
218          <1> loc_ccdrv_reset_cdir_FAT_12_16_fcluster:
219          <1> ;xor     al, al ; xor eax, eax
220          <1> ; 26/07/2022
221 000077C7 31C0          <1>    xor     eax, eax
222          <1> ;xor     edx, edx
223          <1> loc_ccdrv_check_rootdir_sign:
224 000077C9 80BE8000000000          <1>    cmp     byte [esi+LD_CurrentDirectory], 0
225 000077D0 7510          <1>    jne     short loc_ccdrv_reset_cdir_FAT_fcluster
226          <1> loc_ccdrv_set_rootdir_FAT_fcluster:
227 000077D2 89868C000000          <1>    mov     [esi+LD_CurrentDirectory+12], eax
228 000077D8 C78680000000524F4F-          <1>    mov     dword [esi+LD_CurrentDirectory], 'ROOT'
229 000077E1 54          <1>
230          <1>
231          <1> loc_ccdrv_reset_cdir_FAT_fcluster:
232 000077E2 A3[5C770100]          <1>    mov     [Current_Dir_FCluster], eax
233          <1>
234 000077E7 BF[C07E0100]          <1>    mov     edi, PATH_Array
235 000077EC 89F2          <1>    mov     edx, esi

```

```

235 000077EE 81C680000000 <1> add esi, LD_CurrentDirectory
236 000077F4 B920000000 <1> mov ecx, 32
237 000077F9 F3A5 <1> rep movsd
238 <1>
239 000077FB E8042B0000 <1> call change_prompt_dir_string
240 <1>
241 00007800 89D6 <1> mov esi, edx
242 <1>
243 00007802 29C0 <1> sub eax, eax
244 <1> ;sub edx, edx
245 00007804 BF[63770100] <1> mov edi, Current_Dir_Drv
246 <1>
247 00007809 A2[7E2E0100] <1> mov [Restore_CDIRE], al ; 0
248 0000780E C3 <1> retn
249 <1>
250 <1> dos_prompt:
251 <1> ; 26/07/2022 (TRDOS 386 kernel v2.0.5)
252 <1> ; 06/05/2016
253 <1> ; 30/01/2016
254 <1> ; 29/01/2016
255 <1> ; 16/01/2016 (TRDOS 386 = TRDOS v2.0)
256 <1> ; 15/09/2011
257 <1> ; 13/09/2009
258 <1> ; 2004-2005
259 <1>
260 <1> ; 06/05/2016
261 0000780F C705[20830100]- <1> mov dword [mainprog_return_addr], return_from_cmd_interpreter
261 00007815 [C1780000] <1>
262 <1>
263 <1> loc_TRDOS_prompt:
264 00007819 BF[62780100] <1> mov edi, TextBuffer
265 0000781E C6075B <1> mov byte [edi], "["
266 00007821 47 <1> inc edi
267 00007822 BE[DA2E0100] <1> mov esi, TRDOSPromptLabel
268 <1> get_next_prompt_label_char:
269 00007827 803E20 <1> cmp byte [esi], 20h
270 0000782A 7203 <1> jb short pass_prompt_label
271 0000782C A4 <1> movsb
272 0000782D EBF8 <1> jmp short get_next_prompt_label_char
273 <1> pass_prompt_label:
274 0000782F C6075D <1> mov byte [edi], "]"
275 00007832 47 <1> inc edi
276 00007833 C60720 <1> mov byte [edi], 20h
277 00007836 47 <1> inc edi
278 00007837 BE[63770100] <1> mov esi, Current_Dir_Drv
279 0000783C 66A5 <1> movsw
280 0000783E A4 <1> movsb
281 <1> loc_prompt_current_directory:
282 0000783F 803E20 <1> cmp byte [esi], 20h
283 00007842 7203 <1> jb short pass_prompt_current_directory
284 00007844 A4 <1> movsb
285 00007845 EBF8 <1> jmp short loc_prompt_current_directory
286 <1> pass_prompt_current_directory:
287 00007847 C6073E <1> mov byte [edi], '>'
288 0000784A 47 <1> inc edi
289 0000784B C60700 <1> mov byte [edi], 0
290 0000784E BE[62780100] <1> mov esi, TextBuffer
291 00007853 E852F5FFFF <1> call print_msg
292 <1>
293 <1> ;sub bh, bh ; video page = 0
294 <1> ;call get_cpos ; get cursor position
295 00007858 668B15[BE760100] <1> mov dx, [CURSOR_POSN] ; video page 0
296 0000785F 8815[C2770100] <1> mov [CursorColumn], dl
297 <1>
298 <1> ; 30/01/2016 (to show cursor on the row, again)
299 <1> ; (Initial color attributes of video page 0 is 0)
300 <1> ; (see: 'StartPMP' in trdos386.s)
301 <1> ;
302 <1> ;mov edi, 0B8000h ; start of video page 0
303 <1> ;movzx ecx, dl ; column
304 <1> ;mov al, 80
305 <1> ;mul dh
306 <1> ;add ax, cx
307 <1> ;shl ax, 1 ; character + attribute
308 <1> ;add di, ax ; (2*80*row) + (2*column)
309 <1> ;neg cl
310 <1> ;add cl, 80
311 <1> ;mov ax, 700h ; ah = 7 (color attribute)
312 <1> ;rep stosw
313 <1>
314 <1> loc_rw_char:
315 00007865 E8AB000000 <1> call rw_char
316 <1> loc_move_command:
317 0000786A BE[12780100] <1> mov esi, CommandBuffer
318 0000786F 89F7 <1> mov edi, esi
319 00007871 31C9 <1> xor ecx, ecx
320 <1> first_command_char:
321 00007873 AC <1> lodsb
322 00007874 3C20 <1> cmp al, 20h
323 00007876 770C <1> ja short pass_space_control
324 00007878 723F <1> jb short loc_move_cmd_arguments_ok
325 0000787A 81FE[61780100] <1> cmp esi, CommandBuffer + 79
326 00007880 72F1 <1> jb short first_command_char
327 00007882 EB35 <1> jmp short loc_move_cmd_arguments_ok
328 <1>
329 <1> ; 26/07/2022
330 <1> pass_space_control:
331 00007884 3C61 <1> cmp al, 61h
332 00007886 7206 <1> jb short pass_capitalize
333 00007888 3C7A <1> cmp al, 7Ah
334 0000788A 7702 <1> ja short pass_capitalize
335 0000788C 24DF <1> and al, 0DFh
336 <1> pass_capitalize:
337 0000788E AA <1> stosb
338 0000788F FEC1 <1> inc cl
339 00007891 81FE[61780100] <1> cmp esi, CommandBuffer + 79
340 <1> ;jb short next_command_char
341 <1> ; 26/07/2022
342 00007897 7320 <1> jnb short loc_move_cmd_arguments_ok
343 <1>
344 <1> next_command_char:
345 00007899 AC <1> lodsb
346 0000789A 3C20 <1> cmp al, 20h
347 0000789C 77E6 <1> ja short pass_space_control
348 0000789E 7219 <1> jb short loc_move_cmd_arguments_ok
349 <1>
350 <1> loc_1st_cmd_arg: ; 30/01/2016
351 000078A0 AC <1> lodsb
352 000078A1 3C20 <1> cmp al, 20h
353 000078A3 74FB <1> je short loc_1st_cmd_arg
354 000078A5 7212 <1> jb short loc_move_cmd_arguments_ok
355 <1>
356 000078A7 C60700 <1> mov byte [edi], 0
357 000078AA 47 <1> inc edi

```

```

358
359
360 000078AB AA
361 000078AC 81FE[61780100]
362 000078B2 7305
363 000078B4 AC
364 000078B5 3C20
365 000078B7 73F2
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383 000078B9 C60700
384
385
386 000078BC E8A8070000
387
388
389 000078C1 B950000000
390
391 000078C6 BF[12780100]
392 000078CB 30C0
393 000078CD F3AA
394
395
396
397
398 000078CF 803D[1E680000]03
399 000078D6 7419
400
401 000078D8 E8AAA2FFFF
402
403 000078DD 30C0
404
405
406
407 000078DF 3805[CE760100]
408
409
410 000078E5 7405
411
412
413 000078E7 E8C3A6FFFF
414
415 000078EC E928FFFFFF
416
417
418 000078F1 BE[07390100]
419 000078F6 E8AFF4FFFF
420 000078FB EBE2
421
422
423
424
425 000078FD 80FC4B
426 00007900 7426
427 00007902 80FC53
428 00007905 7421
429 00007907 80FC4D
430 0000790A 7509
431 0000790C 80FA4F
432 0000790F 7304
433 00007911 FEC2
434 00007913 EB1D
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450 00007915 30E4
451 00007917 E80096FFFF
452 0000791C 20C0
453 0000791E 74DD
454 00007920 3CE0
455 00007922 74D9
456 00007924 3C08
457 00007926 752A
458
459 00007928 3A15[C2770100]
460 0000792E 76E5
461
462 00007930 FECA
463
464
465 00007932 52
466
467
468 00007933 E83BAAFFFF
469 00007938 5A
470
471
472 00007939 88D3
473 0000793B 2A1D[C2770100]
474 00007941 B020
475 00007943 8883[12780100]
476
477
478 00007949 B307
479
480 0000794B E80AA9FFFF
481

```

```

<1>
<1> loc_move_cmd_arguments:
<1> stosb
<1> cmp esi, CommandBuffer + 79
<1> jnb short loc_move_cmd_arguments_ok
<1> lodsb
<1> cmp al, 20h
<1> jnb short loc_move_cmd_arguments
<1> ; 26/07/2022
<1> ; jmp short loc_move_cmd_arguments_ok
<1>
<1> ; 26/07/2022
<1> ; pass_space_control:
<1> ; cmp al, 61h
<1> ; jnb short pass_capitalize
<1> ; cmp al, 7Ah
<1> ; ja short pass_capitalize
<1> ; and al, 0Dh
<1> ; pass_capitalize:
<1> ; stosb
<1> ; inc cl
<1> ; cmp esi, CommandBuffer + 79
<1> ; jnb short next_command_char
<1>
<1> loc_move_cmd_arguments_ok:
<1> mov byte [edi], 0
<1>
<1> call_command_interpreter:
<1> call command_interpreter
<1>
<1> return_from_cmd_interpreter:
<1> mov ecx, 80
<1> ; mov cx, 80
<1> mov edi, CommandBuffer
<1> xor al, al
<1> rep stosb
<1> ; cmp byte [Program_Exit], 0
<1> ; ja short loc_terminate_trdos
<1>
<1> ; 16/01/2016
<1> cmp byte [CRT_MODE], 3 ; 80*25 color
<1> je short pass_set_txt_mode
<1>
<1> call set_txt_mode ; set vide mode to 03h
<1> ; 07/01/2017
<1> xor al, al
<1>
<1> loc_check_active_page:
<1> ; xor al, al
<1> cmp [ACTIVE_PAGE], al ; 0
<1> ; je loc_TRDOS_prompt
<1> ; 26/07/2022
<1> je short loc_prompt_again
<1>
<1> ; AL = 0 = video page 0
<1> call set_active_page
<1> loc_prompt_again: ; 26/07/2022
<1> jmp loc_TRDOS_prompt ; infinitive loop
<1>
<1> pass_set_txt_mode:
<1> mov esi, nextline
<1> call print_msg
<1> jmp short loc_check_active_page
<1>
<1> ; rw_char:
<1> ; 26/07/2022 (TRDOS 386 Kernel v2.0.5)
<1> loc_arrow:
<1> cmp ah, 4Bh
<1> je short loc_back
<1> cmp ah, 53h
<1> je short loc_back
<1> cmp ah, 4Dh
<1> jne short readnextchar
<1> cmp dl, 79
<1> jnb short readnextchar
<1> inc dl
<1> jmp short set_cursor_pos
<1>
<1> rw_char:
<1> ; 26/07/2022 (TRDOS 386 Kernel v2.0.5)
<1> ; 13/05/2016
<1> ; 30/01/2016
<1> ; 29/01/2016
<1> ; 17/01/2016 (TRDOS 386 = TRDOS v2.0)
<1> ; 2004-2005
<1>
<1> ; DH = cursor row, DL = cursor column
<1> ; BH = 0 = video page number (active page)
<1>
<1> ; xor bh, bh ; 0 = video page 0
<1>
<1> readnextchar:
<1> xor ah, ah
<1> call int16h
<1> and al, al
<1> jz short loc_arrow
<1> cmp al, 0E0h
<1> je short loc_arrow
<1> cmp al, 08h
<1> jne short char_return
<1> loc_back:
<1> cmp dl, [CursorColumn]
<1> jna short readnextchar
<1> prev_column:
<1> dec dl
<1> set_cursor_pos:
<1> ; push dx
<1> push edx ; 29/12/2017
<1> ; xor bh, bh ; 0 = video page 0
<1> ; DH = Row, DL = Column
<1> call _set_cpos ; 17/01/2016
<1> pop edx ; 29/12/2017
<1> ; pop dx
<1> ; movzx ebx, dl
<1> mov bl, dl
<1> sub bl, [CursorColumn]
<1> mov al, 20h
<1> mov [CommandBuffer+ebx], al
<1> ; sub bh, bh ; video page 0
<1> ; mov cx, 1
<1> mov bl, 7 ; color attribute
<1> ; bh = 0 ; 26/07/2022
<1> call _write_c_current ; 17/01/2016
<1> ; mov dx, [CURSOR_POSN]

```

```

482 00007950 EBC3      <1>      jmp      short readnextchar
483                  <1>
484                  <1>      ; 26/07/2022
485                  <1> ;loc_arrow:
486                  <1> ;      cmp      ah, 4Bh
487                  <1> ;      je       short loc_back
488                  <1> ;      cmp      ah, 53h
489                  <1> ;      je       short loc_back
490                  <1> ;      cmp      ah, 4Dh
491                  <1> ;      jne      short readnextchar
492                  <1> ;      cmp      dl, 79
493                  <1> ;      jnb      short readnextchar
494                  <1> ;      inc      dl
495                  <1> ;      jmp      short set_cursor_pos
496                  <1>
497                  <1> char_return:
498 00007952 0FB6DA      <1>      movzx   ebx, dl
499 00007955 2A1D[C2770100] <1>      sub      bl, [CursorColumn]
500 0000795B 3C20          <1>      cmp      al, 20h
501 0000795D 721D          <1>      jb       short loc_escape
502 0000795F 8883[12780100] <1>      mov      [CommandBuffer+ebx], al
503 00007965 80FA4F        <1>      cmp      dl, 79
504 00007968 73AB          <1>      jnb      short readnextchar
505 0000796A 66BB0700     <1>      mov      bx, 7 ; color attribute
506 0000796E E868A9FFFF <1>      call     _write_tty
507 00007973 668B15[BE760100] <1>      mov      dx, [CURSOR_POSN] ; video page 0
508 0000797A EB99          <1>      jmp      short readnextchar ; 26/07/2022
509                  <1>
510                  <1> loc_escape:
511 0000797C 3C1B          <1>      cmp      al, 1Bh
512 0000797E 7414          <1>      je       short rw_char_retn
513                  <1> ;
514 00007980 3C0D          <1>      cmp      al, 0Dh ; CR
515 00007982 7591          <1>      jne      short readnextchar ; 26/07/2022
516                  <1>
517                  <1> ; 13/05/2016
518 00007984 66BB0700     <1>      mov      bx, 7 ; attribute/color (bl)
519                  <1> ;      ; video page 0 (bh=0)
520 00007988 E84EA9FFFF <1>      call     _write_tty
521                  <1> ;mov      bx, 7 ; attribute/color
522                  <1> ;      ; video page 0 (bh=0)
523 0000798D B00A          <1>      mov      al, 0Ah ; LF
524 0000798F E847A9FFFF <1>      call     _write_tty
525                  <1> rw_char_retn:
526 00007994 C3          <1>      retn
527                  <1>
528                  <1> show_date:
529                  <1> ; 26/07/2022 (TRDOS 386 Kernel v2.0.5)
530                  <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
531                  <1> ; 2004-2005
532                  <1>
533                  <1> ;mov      ah, 04h
534                  <1> ;call     int1Ah
535 00007995 E822EAFFFF <1>      call     RTC_40 ; GET RTC DATE
536                  <1>
537 0000799A 88D0          <1>      mov      al, dl
538 0000799C E86E95FFFF <1>      call     bcd_to_ascii
539 000079A1 66A3[C62F0100] <1>      mov      [Day], ax
540                  <1>
541 000079A7 88F0          <1>      mov      al, dh
542 000079A9 E86195FFFF <1>      call     bcd_to_ascii
543 000079AE 66A3[C92F0100] <1>      mov      [Month], ax
544                  <1>
545 000079B4 88E8          <1>      mov      al, ch
546 000079B6 E85495FFFF <1>      call     bcd_to_ascii
547 000079BB 66A3[CC2F0100] <1>      mov      [Century], ax
548                  <1>
549 000079C1 88C8          <1>      mov      al, cl
550 000079C3 E84795FFFF <1>      call     bcd_to_ascii
551 000079C8 66A3[CE2F0100] <1>      mov      [Year], ax
552                  <1>
553 000079CE BE[B62F0100] <1>      mov      esi, Msg_Show_Date
554                  <1> ;call     print_msg
555                  <1> ;retn
556                  <1> ; 26/07/2022
557 000079D3 E9D2F3FFFF <1>      jmp      print_msg
558                  <1>
559                  <1> set_date:
560                  <1> ; 26/07/2022 (TRDOS 386 Kernel v2.0.5)
561                  <1> ; 13/05/2016
562                  <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
563                  <1> ; 2004-2005
564                  <1>
565 000079D8 BE[9A2F0100] <1>      mov      esi, Msg_Enter_Date
566 000079DD E8C8F3FFFF <1>      call     print_msg
567                  <1>
568                  <1> loc_enter_day_1:
569 000079E2 30E4          <1>      xor      ah, ah
570 000079E4 E83395FFFF <1>      call     int16h
571                  <1> ; AL = ASCII Code of the Character
572 000079E9 3C0D          <1>      cmp      al, 13
573                  <1> ;je       loc_set_date_retn
574                  <1> ; 26/07/2022
575 000079EB 7404          <1>      je       short set_date_0
576 000079ED 3C1B          <1>      cmp      al, 27
577                  <1> ;je       loc_set_date_retn
578                  <1> ; 26/07/2022
579 000079EF 7511          <1>      jne      short set_date_1
580                  <1>
581                  <1> set_date_0:
582                  <1> ;loc_set_date_retn:
583 000079F1 BE[07390100] <1>      mov      esi, nextline
584                  <1> ;call     print_msg
585                  <1> ;retn
586                  <1> ; 26/07/2022
587 000079F6 E9AFF3FFFF <1>      jmp      print_msg
588                  <1>
589                  <1> ; 26/07/2022
590                  <1> loc_set_date_stc_0:
591                  <1> ;xor      bh, bh ; video page 0
592 000079FB E8C5A9FFFF <1>      call     beeper ; BEEP !
593 00007A00 EBE0          <1>      jmp      short loc_enter_day_1
594                  <1>
595                  <1> ; 26/07/2022
596                  <1> set_date_1:
597 00007A02 A2[C62F0100] <1>      mov      [Day], al
598 00007A07 3C30          <1>      cmp      al, '0'
599 00007A09 72F0          <1>      jb       short loc_set_date_stc_0
600 00007A0B 3C33          <1>      cmp      al, '3'
601 00007A0D 77EC          <1>      ja       short loc_set_date_stc_0
602                  <1> ; 13/05/2016
603                  <1> ;mov      bx, 7 ; attribute/color (bl)
604                  <1> ;      ; video page 0 (bh)
605 00007A0F B307          <1>      mov      bl, 7

```

```

606 00007A11 E8C5A8FFFF <1> call _write_tty
607 <1> loc_enter_day_2:
608 00007A16 30E4 <1> xor ah, ah
609 00007A18 E8FF94FFFF <1> call int16h
610 <1> ; AL = ASCII Code of the Character
611 00007A1D 3C1B <1> cmp al, 27
612 <1> ;je loc_set_date_retn
613 <1> ; 26/07/2022
614 00007A1F 74D0 <1> je short set_date_0
615 00007A21 A2[C72F0100] <1> mov [Day+1], al
616 00007A26 3C30 <1> cmp al, '0'
617 00007A28 7211 <1> jb short loc_set_date_stc_1
618 00007A2A 3C39 <1> cmp al, '9'
619 00007A2C 770D <1> ja short loc_set_date_stc_1
620 00007A2E 803D[C62F0100]33 <1> cmp byte [Day], '3'
621 00007A35 7219 <1> jb short pass_set_day_31
622 00007A37 3C31 <1> cmp al, '1'
623 <1> ;ja loc_set_date_stc_1
624 <1> ; 26/07/2022
625 00007A39 7615 <1> jna short pass_set_day_31
626 <1>
627 <1> ; 26/07/2022
628 <1> loc_set_date_stc_1:
629 00007A3B E8E4010000 <1> call check_for_backspace
630 00007A40 7407 <1> je short loc_set_date_bs_1
631 <1> ;xor bh, bh ; video page 0
632 00007A42 E87EA9FFFF <1> call beeper ; BEEP !
633 00007A47 EBCD <1> jmp short loc_enter_day_2
634 <1> loc_set_date_bs_1:
635 00007A49 E8C4010000 <1> call write_backspace
636 00007A4E EB92 <1> jmp short loc_enter_day_1
637 <1>
638 <1> pass_set_day_31:
639 <1> ; 13/05/2016
640 <1> ;mov bx, 7 ; attribute/color (b1)
641 <1> ; ; video page 0 (bh)
642 00007A50 B307 <1> mov bl, 7
643 00007A52 E884A8FFFF <1> call _write_tty
644 <1> loc_enter_separator_1:
645 00007A57 28E4 <1> sub ah, ah ; 0
646 00007A59 E8BE94FFFF <1> call int16h
647 <1> ; AL = ASCII Code of the Character
648 00007A5E 3C1B <1> cmp al, 27
649 <1> ;je loc_set_date_retn
650 <1> ; 26/07/2022
651 00007A60 748F <1> je short set_date_0
652 00007A62 3C2D <1> cmp al, '-'
653 00007A64 7443 <1> je short pass_set_date_separator_1
654 00007A66 3C2F <1> cmp al, '/'
655 <1> ;jne loc_set_date_stc_2
656 <1> ; 26/07/2022
657 00007A68 743F <1> je short pass_set_date_separator_1
658 <1>
659 <1> ; 26/07/2022
660 <1> loc_set_date_stc_2:
661 00007A6A E8B5010000 <1> call check_for_backspace
662 00007A6F 7407 <1> je short loc_set_date_bs_2
663 <1> ;xor bh, bh ; video page 0
664 00007A71 E84FA9FFFF <1> call beeper ; BEEP !
665 00007A76 EBD5 <1> jmp short loc_enter_separator_1
666 <1> loc_set_date_bs_2:
667 00007A78 E895010000 <1> call write_backspace
668 00007A7D EB97 <1> jmp short loc_enter_day_2
669 <1>
670 <1> ; 26/07/2022
671 <1> loc_set_date_stc_3:
672 00007A7F E8A0010000 <1> call check_for_backspace
673 00007A84 7407 <1> je short loc_set_date_bs_3
674 <1> ;xor bh, bh ; video page 0
675 00007A86 E83AA9FFFF <1> call beeper ; BEEP !
676 00007A8B EB23 <1> jmp short loc_enter_month_1
677 <1> loc_set_date_bs_3:
678 00007A8D E880010000 <1> call write_backspace
679 00007A92 EBC3 <1> jmp short loc_enter_separator_1
680 <1>
681 <1> ; 26/07/2022
682 <1> loc_set_date_stc_4:
683 00007A94 E88B010000 <1> call check_for_backspace
684 00007A99 7407 <1> je short loc_set_date_bs_4
685 <1> ;xor bh, bh ; video page 0
686 00007A9B E825A9FFFF <1> call beeper ; BEEP !
687 00007AA0 EB2D <1> jmp short loc_enter_month_2
688 <1> loc_set_date_bs_4:
689 00007AA2 E86B010000 <1> call write_backspace
690 00007AA7 EB07 <1> jmp short loc_enter_month_1
691 <1>
692 <1> pass_set_date_separator_1:
693 <1> ; 13/05/2016
694 <1> ;mov bx, 7 ; attribute/color (b1)
695 <1> ; ; video page 0 (bh)
696 00007AA9 B307 <1> mov bl, 7
697 00007AAB E82BA8FFFF <1> call _write_tty
698 <1> loc_enter_month_1:
699 00007AB0 30E4 <1> xor ah, ah ; 0
700 00007AB2 E86594FFFF <1> call int16h
701 <1> ; AL = ASCII Code of the Character
702 00007AB7 3C1B <1> cmp al, 27
703 <1> ;je loc_set_date_retn
704 <1> ; 26/07/2022
705 <1> ;je short loc_set_date_ok
706 <1> ; 07/08/2022
707 00007AB9 741F <1> je short jmp_loc_set_date_ok
708 00007ABB A2[C92F0100] <1> mov [Month], al
709 00007AC0 3C30 <1> cmp al, '0'
710 00007AC2 72BB <1> jb short loc_set_date_stc_3
711 00007AC4 3C31 <1> cmp al, '1'
712 00007AC6 77B7 <1> ja short loc_set_date_stc_3
713 <1> ; 13/05/2016
714 <1> ;mov bx, 7 ; attribute/color (b1)
715 <1> ; ; video page 0 (bh)
716 00007AC8 B307 <1> mov bl, 7
717 00007ACA E80CA8FFFF <1> call _write_tty
718 <1> loc_enter_month_2:
719 00007ACF 30E4 <1> xor ah, ah
720 00007AD1 E84694FFFF <1> call int16h
721 <1> ; AL = ASCII Code of the Character
722 00007AD6 3C1B <1> cmp al, 27
723 <1> ;je loc_set_date_retn
724 <1> ; 26/07/2022
725 <1> ;je short loc_set_date_ok
726 <1> ; 07/08/2022
727 00007AD8 7505 <1> jne short loc_enter_month_3
728 <1> jmp_loc_set_date_ok:
729 00007ADA E996000000 <1> jmp loc_set_date_ok

```



```

730      <1> loc_enter_month_3:
731 00007ADF A2[CA2F0100] <1>      mov     [Month+1], al
732 00007AE4 3C30 <1>      cmp     al, '0'
733 00007AE6 72AC <1>      jnb     short loc_set_date_stc_4
734 00007AE8 3C39 <1>      cmp     al, '9'
735 00007AEA 77A8 <1>      ja      short loc_set_date_stc_4
736 00007AEC 803D[C92F0100]31 <1>      cmp     byte [Month], '1'
737 00007AF3 7204 <1>      jnb     short pass_set_month_12
738 00007AF5 3C32 <1>      cmp     al, '2'
739 00007AF7 779B <1>      ja      short loc_set_date_stc_4
740      <1> pass_set_month_12:
741      <1>      ; 13/05/2016
742      <1>      ;mov     bx, 7 ; attribute/color (b1)
743      <1>      ; video page 0 (bh)
744 00007AF9 B307 <1>      mov     bl, 7
745 00007AFB E8DBA7FFFF <1>      call    _write_tty
746      <1> loc_enter_separator_2:
747 00007B00 28E4 <1>      sub     ah, ah
748 00007B02 E81594FFFF <1>      call    int16h
749      <1>      ; AL = ASCII Code of the Character
750 00007B07 3C1B <1>      cmp     al, 27
751      <1>      ;je loc_set_date_retn
752      <1>      ; 26/07/2022
753 00007B09 746A <1>      je      short loc_set_date_ok
754 00007B0B 3C2D <1>      cmp     al, '-'
755 00007B0D 7404 <1>      je      short pass_set_date_separator_2
756 00007B0F 3C2F <1>      cmp     al, '/'
757 00007B11 756C <1>      jne     short loc_set_date_stc_5
758      <1> pass_set_date_separator_2:
759      <1>      ; 13/05/2016
760      <1>      ;mov     bx, 7 ; attribute/color (b1)
761      <1>      ; video page 0 (bh)
762 00007B13 B307 <1>      mov     bl, 7
763 00007B15 E8C1A7FFFF <1>      call    _write_tty
764      <1> loc_enter_year_1:
765 00007B1A 30E4 <1>      xor     ah, ah
766 00007B1C E8FB93FFFF <1>      call    int16h
767      <1>      ; AL = ASCII Code of the Character
768 00007B21 3C1B <1>      cmp     al, 27
769      <1>      ;je loc_set_date_retn
770      <1>      ; 26/07/2022
771 00007B23 7450 <1>      je      short loc_set_date_ok
772 00007B25 A2[CE2F0100] <1>      mov     [Year], al
773 00007B2A 3C30 <1>      cmp     al, '0'
774 00007B2C 726C <1>      jnb     short loc_set_date_stc_6
775 00007B2E 3C39 <1>      cmp     al, '9'
776 00007B30 7768 <1>      ja      short loc_set_date_stc_6
777      <1>      ; 13/05/2016
778      <1>      ;mov     bx, 7 ; attribute/color (b1)
779      <1>      ; video page 0 (bh)
780 00007B32 B307 <1>      mov     bl, 7
781 00007B34 E8A2A7FFFF <1>      call    _write_tty
782      <1> loc_enter_year_2:
783 00007B39 30E4 <1>      xor     ah, ah
784 00007B3B E8DC93FFFF <1>      call    int16h
785      <1>      ; AL = ASCII Code of the Character
786 00007B40 3C1B <1>      cmp     al, 27
787      <1>      ;je loc_set_date_retn
788      <1>      ; 26/07/2022
789 00007B42 7431 <1>      je      short loc_set_date_ok
790 00007B44 A2[CF2F0100] <1>      mov     byte [Year+1], al
791 00007B49 3C30 <1>      cmp     al, '0'
792 00007B4B 7268 <1>      jnb     short loc_set_date_stc_7
793 00007B4D 3C39 <1>      cmp     al, '9'
794 00007B4F 7764 <1>      ja      short loc_set_date_stc_7
795      <1>      ; 13/05/2016
796      <1>      ;mov     bx, 7 ; attribute/color (b1)
797      <1>      ; video page 0 (bh)
798 00007B51 B307 <1>      mov     bl, 7
799 00007B53 E883A7FFFF <1>      call    _write_tty
800      <1> loc_set_date_get_lchar_again:
801 00007B58 28E4 <1>      sub     ah, ah ; 0
802 00007B5A E8BD93FFFF <1>      call    int16h
803      <1>      ; AL = ASCII Code of the Character
804 00007B5F 3C0D <1>      cmp     al, 13 ; ENTER key
805 00007B61 746D <1>      je      short loc_set_date_progress
806 00007B63 3C1B <1>      cmp     al, 27 ; ESC key
807      <1>      ;je loc_set_date_retn
808      <1>      ; 26/07/2022
809 00007B65 740E <1>      je      short loc_set_date_ok
810      <1>      ;
811 00007B67 E8B8000000 <1>      call    check_for_backspace
812 00007B6C 75EA <1>      jne     short loc_set_date_get_lchar_again
813      <1>
814      <1> loc_set_date_bs_8:
815 00007B6E E89F000000 <1>      call    write_backspace
816 00007B73 EBC4 <1>      jmp     short loc_enter_year_2
817      <1>
818      <1> loc_set_date_ok:
819      <1> ;loc_set_date_retn:
820 00007B75 BE[07390100] <1>      mov     esi, nextline
821      <1>      ;call    print_msg
822      <1>      ;retn
823      <1>      ; 26/07/2022
824 00007B7A E92BF2FFFF <1>      jmp     print_msg
825      <1>
826      <1>      ; 26/07/2022
827      <1> ;loc_set_date_stc_0:
828      <1> ;xor     bh, bh ; video page 0
829      <1> ;call    beeper ; BEEP !
830      <1> ;jmp     loc_enter_day_1
831      <1> ;loc_set_date_stc_1:
832      <1> ;call    check_for_backspace
833      <1> ;je      short loc_set_date_bs_1
834      <1> ;xor     bh, bh ; video page 0
835      <1> ;call    beeper ; BEEP !
836      <1> ;jmp     loc_enter_day_2
837      <1> ;loc_set_date_bs_1:
838      <1> ;call    write_backspace
839      <1> ;jmp     loc_enter_day_1
840      <1> ;loc_set_date_stc_2:
841      <1> ;call    check_for_backspace
842      <1> ;je      short loc_set_date_bs_2
843      <1> ;xor     bh, bh ; video page 0
844      <1> ;call    beeper ; BEEP !
845      <1> ;jmp     loc_enter_separator_1
846      <1> ;loc_set_date_bs_2:
847      <1> ;call    write_backspace
848      <1> ;jmp     loc_enter_day_2
849      <1> ;loc_set_date_stc_3:
850      <1> ;call    check_for_backspace
851      <1> ;je      short loc_set_date_bs_3
852      <1> ;xor     bh, bh ; video page 0
853      <1> ;call    beeper ; BEEP !

```

```

854      <1> ; jmp loc_enter_month_1
855      <1> ;loc_set_date_bs_3:
856      <1> ; call write_backspace
857      <1> ; jmp loc_enter_separator_1
858      <1> ;loc_set_date_stc_4:
859      <1> ; call check_for_backspace
860      <1> ; je short loc_set_date_bs_4
861      <1> ; ;xor bh, bh ; video page 0
862      <1> ; call beeper ; BEEP !
863      <1> ; jmp loc_enter_month_2
864      <1> ;loc_set_date_bs_4:
865      <1> ; call write_backspace
866      <1> ; jmp loc_enter_month_1
867      <1>
868      <1> ; 26/07/2022
869      <1> loc_set_date_stc_5:
870      <1> call check_for_backspace
871      <1> je short loc_set_date_bs_5
872      <1> ;xor bh, bh ; video page 0
873      <1> call beeper ; BEEP !
874      <1> jmp loc_enter_separator_2
875      <1> loc_set_date_bs_5:
876      <1> call write_backspace
877      <1> jmp loc_enter_month_2
878      <1> loc_set_date_stc_6:
879      <1> call check_for_backspace
880      <1> je short loc_set_date_bs_6
881      <1> ;xor bh, bh ; video page 0
882      <1> call beeper ; BEEP !
883      <1> jmp loc_enter_year_1
884      <1> loc_set_date_bs_6:
885      <1> call write_backspace
886      <1> jmp loc_enter_separator_2
887      <1> loc_set_date_stc_7:
888      <1> call check_for_backspace
889      <1> je short loc_set_date_bs_7
890      <1> ;xor bh, bh ; video page 0
891      <1> call beeper ; BEEP !
892      <1> jmp loc_enter_year_2
893      <1> loc_set_date_bs_7:
894      <1> call write_backspace
895      <1> jmp loc_enter_year_1
896      <1>
897      <1> loc_set_date_progress:
898      <1> ; Get Current Date
899      <1> ;mov ah, 04h
900      <1> ;call int1Ah
901      <1> call RTC_40 ; GET RTC DATE
902      <1> ; CH = century (in BCD)
903      <1>
904      <1> mov ax, [Year]
905      <1> sub ax, '00'
906      <1> shl al, 4 ; * 16
907      <1> mov cl, al
908      <1> add cl, ah
909      <1> mov ax, [Month]
910      <1> sub ax, '00'
911      <1> shl al, 4 ; * 16
912      <1> mov dh, al
913      <1> add dh, ah
914      <1> mov ax, [Day]
915      <1> sub ax, '00'
916      <1> shl al, 4 ; * 16
917      <1> mov dl, al
918      <1> add dl, ah
919      <1>
920      <1> ;mov ah, 05h
921      <1> ;call int1Ah
922      <1> call RTC_50 ; SET RTC DATE
923      <1>
924      <1> ;loc_set_date_retn:
925      <1> ; mov esi, nextline
926      <1> ; call print_msg
927      <1> ; retn
928      <1> ; 26/07/2022
929      <1> ; jmp print_msg
930      <1>
931      <1> ; 26/07/2022
932      <1> jmp loc_set_date_ok
933      <1>
934      <1> write_backspace:
935      <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
936      <1> mov al, 08h ; BACKSPACE
937      <1> ; 13/05/2016
938      <1> mov bx, 7 ; b1 = attribute/color
939      <1> ; bh = video page = 0
940      <1> call _write_tty
941      <1> mov al, 20h ; BLANK/SPACE char
942      <1> ;mov bx, 7 ; attribute/color
943      <1> ;call _write_c_current
944      <1> ;retn
945      <1> jmp _write_c_current
946      <1>
947      <1> check_for_backspace:
948      <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
949      <1> cmp ax, 0E08h
950      <1> je short cfbs_retn
951      <1> cmp ax, 4BE0h
952      <1> je short cfbs_retn
953      <1> cmp ax, 4B00h
954      <1> je short cfbs_retn
955      <1> cmp ax, 53E0h
956      <1> cfbs_retn:
957      <1> retn
958      <1>
959      <1> show_time:
960      <1> ; 26/07/2022 (TRDOS 386 kernel v2.0.5)
961      <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
962      <1> ; 2004-2005
963      <1>
964      <1> ;mov ah, 02h
965      <1> ;call int1Ah
966      <1> call RTC_20 ; GET RTC TIME
967      <1>
968      <1> mov al, ch
969      <1> call bcd_to_ascii
970      <1> mov [Hour], ax
971      <1>
972      <1> mov al, cl
973      <1> call bcd_to_ascii
974      <1> mov [Minute], ax
975      <1>
976      <1> mov al, dh
977      <1> call bcd_to_ascii

```

```

978 00007C61 66A3[FA2F0100] <1> mov [Second], ax
979 <1>
980 00007C67 BE[E42F0100] <1> mov esi, Msg_Show_Time
981 <1> ;call print_msg
982 <1> ;retn
983 <1> ; 26/07/2022
984 00007C6C E939F1FFFF <1> jmp print_msg
985 <1>
986 <1> set_time:
987 <1> ; 26/07/2022 (TRDOS 386 Kernel v2.0.5)
988 <1> ; 13/05/2016
989 <1> ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
990 <1> ; 2004-2005
991 <1>
992 00007C71 BE[D32F0100] <1> mov esi, Msg_Enter_Time
993 00007C76 E82FF1FFFF <1> call print_msg
994 <1>
995 <1> loc_enter_hour_1:
996 00007C7B 30E4 <1> xor ah, ah
997 00007C7D E89A92FFFF <1> call int16h
998 <1> ; AL = ASCII Code of the Character
999 00007C82 3C0D <1> cmp al, 13 ; ENTER key
1000 00007C84 742D <1> je short loc_set_time_retn
1001 00007C86 3C1B <1> cmp al, 27 ; ESC key
1002 00007C88 7429 <1> je short loc_set_time_retn
1003 <1> set_time_0:
1004 00007C8A A2[F42F0100] <1> mov [Hour], al
1005 00007C8F 3C30 <1> cmp al, '0'
1006 00007C91 7204 <1> jb short loc_set_time_stc_0
1007 00007C93 3C32 <1> cmp al, '2'
1008 <1> ;ja loc_set_time_stc_0
1009 <1> ; 26/07/2022
1010 00007C95 7626 <1> jna short set_time_1
1011 <1>
1012 <1> ; 26/07/2022
1013 <1> loc_set_time_stc_0:
1014 <1> ;xor bh, bh ; video page 0
1015 00007C97 E829A7FFFF <1> call beeper ; BEEP !
1016 00007C9C EBDD <1> jmp short loc_enter_hour_1
1017 <1>
1018 <1> loc_set_time_stc_1:
1019 00007C9E E881FFFFFF <1> call check_for_backspace
1020 00007CA3 7407 <1> je short loc_set_time_bs_1
1021 <1> ;xor bh, bh ; video page 0
1022 00007CA5 E81BA7FFFF <1> call beeper ; BEEP !
1023 00007CAA EB18 <1> jmp short loc_enter_hour_2
1024 <1> loc_set_time_bs_1:
1025 00007CAC E861FFFFFF <1> call write_backspace
1026 00007CB1 EBC8 <1> jmp short loc_enter_hour_1
1027 <1>
1028 <1> ; 26/07/2022
1029 <1> loc_set_time_retn:
1030 00007CB3 BE[07390100] <1> mov esi, nextline
1031 <1> ;call print_msg
1032 <1> ;retn
1033 00007CB8 E9EDF0FFFF <1> jmp print_msg
1034 <1>
1035 <1> set_time_1:
1036 <1> ; 13/05/2016
1037 <1> ;mov bx, 7 ; attribute/color (b1)
1038 <1> ; video page 0 (bh)
1039 00007CBD B307 <1> mov bl, 7
1040 00007CBF E817A6FFFF <1> call _write_tty
1041 <1> loc_enter_hour_2:
1042 00007CC4 30E4 <1> xor ah, ah
1043 00007CC6 E85192FFFF <1> call int16h
1044 <1> ; AL = ASCII Code of the Character
1045 00007CCB 3C1B <1> cmp al, 27
1046 00007CCD 74E4 <1> je short loc_set_time_retn
1047 00007CCF A2[F52F0100] <1> mov [Hour+1], al
1048 00007CD4 3C30 <1> cmp al, '0'
1049 00007CD6 72C6 <1> jb short loc_set_time_stc_1
1050 00007CD8 3C39 <1> cmp al, '9'
1051 00007CDA 77C2 <1> ja short loc_set_time_stc_1
1052 00007CDC 803D[F42F0100]32 <1> cmp byte [Hour], '2'
1053 00007CE3 7204 <1> jb short pass_set_time_24
1054 00007CE5 3C34 <1> cmp al, '4'
1055 00007CE7 77B5 <1> ja short loc_set_time_stc_1
1056 <1> pass_set_time_24:
1057 <1> ; 13/05/2016
1058 <1> ;mov bx, 7 ; attribute/color (b1)
1059 <1> ; video page 0 (bh)
1060 00007CE9 B307 <1> mov bl, 7
1061 00007CEB E8EBA5FFFF <1> call _write_tty
1062 <1> loc_enter_time_separator_1:
1063 00007CF0 28E4 <1> sub ah, ah ; 0
1064 00007CF2 E82592FFFF <1> call int16h
1065 <1> ; AL = ASCII Code of the Character
1066 00007CF7 3C1B <1> cmp al, 27
1067 00007CF9 74B8 <1> je short loc_set_time_retn
1068 00007CFB 3C3A <1> cmp al, '.'
1069 <1> ;jne loc_set_time_stc_2
1070 <1> ; 26/07/2022
1071 00007CFD 7415 <1> je short set_time_2
1072 <1>
1073 <1> ; 26/07/2022
1074 <1> loc_set_time_stc_2:
1075 00007CFF E820FFFFFF <1> call check_for_backspace
1076 00007D04 7407 <1> je short loc_set_time_bs_2
1077 <1> ;xor bh, bh ; video page 0
1078 00007D06 E8BAA6FFFF <1> call beeper ; BEEP !
1079 00007D0B EBE3 <1> jmp short loc_enter_time_separator_1
1080 <1> loc_set_time_bs_2:
1081 00007D0D E800FFFFFF <1> call write_backspace
1082 00007D12 EBB0 <1> jmp short loc_enter_hour_2
1083 <1>
1084 <1> set_time_2:
1085 <1> ; 13/05/2016
1086 <1> ;mov bx, 7 ; attribute/color (b1)
1087 <1> ; video page 0 (bh)
1088 00007D14 B307 <1> mov bl, 7
1089 00007D16 E8C0A5FFFF <1> call _write_tty
1090 <1> loc_enter_minute_1:
1091 00007D1B 30E4 <1> xor ah, ah
1092 00007D1D E8FA91FFFF <1> call int16h
1093 <1> ; AL = ASCII Code of the Character
1094 00007D22 3C1B <1> cmp al, 27
1095 00007D24 748D <1> je short loc_set_time_retn
1096 00007D26 A2[F72F0100] <1> mov [Minute], al
1097 00007D2B 3C30 <1> cmp al, '0'
1098 00007D2D 7204 <1> jb short loc_set_time_stc_3
1099 00007D2F 3C35 <1> cmp al, '5'
1100 <1> ;ja loc_set_time_stc_3
1101 <1> ; 26/07/2022

```

```

1102 00007D31 7615      <1>      jna      short set_time_3
1103                  <1>
1104                  <1>      ; 26/07/2022
1105                  <1> loc_set_time_stc_3:
1106 00007D33 E8ECFEFFFF <1>      call    check_for_backspace
1107 00007D38 7407      <1>      je      short loc_set_time_bs_3
1108                  <1>      ;xor      bh, bh ; video page 0
1109 00007D3A E886A6FFFF <1>      call    beeper ; BEEP !6
1110 00007D3F EBDA      <1>      jmp     short loc_enter_minute_1
1111                  <1> loc_set_time_bs_3:
1112 00007D41 E8CCFEFFFF <1>      call    write_backspace
1113 00007D46 EBA8      <1>      jmp     short loc_enter_time_separator_1
1114                  <1>
1115                  <1> set_time_3:
1116                  <1>      ; 13/05/2016
1117                  <1>      ;mov     bx, 7 ; attribute/color (b1)
1118                  <1>      ; video page 0 (bh)
1119 00007D48 B307      <1>      mov     b1, 7
1120 00007D4A E88CA5FFFF <1>      call    _write_tty
1121                  <1> loc_enter_minute_2:
1122 00007D4F 30E4      <1>      xor     ah, ah
1123 00007D51 E8C691FFFF <1>      call    int16h
1124                  <1>      ; AL = ASCII Code of the Character
1125 00007D56 3C1B      <1>      cmp     al, 27
1126                  <1>      ;je      short loc_set_time_retn
1127                  <1>      ; 07/08/2022
1128 00007D58 7505      <1>      jne     short loc_enter_minute_3
1129 00007D5A E954FFFF   <1>      jmp     loc_set_time_retn
1130                  <1> loc_enter_minute_3:
1131 00007D5F A2[F82F0100] <1>      mov     [Minute+1], al
1132 00007D64 3C30      <1>      cmp     al, '0'
1133 00007D66 7204      <1>      jb      short loc_set_time_stc_4
1134 00007D68 3C39      <1>      cmp     al, '9'
1135                  <1>      ;ja      loc_set_time_stc_4
1136                  <1>      ; 26/07/2022
1137 00007D6A 7615      <1>      jna     short set_time_4
1138                  <1>
1139                  <1>      ; 26/07/2022
1140                  <1> loc_set_time_stc_4:
1141 00007D6C E8B3FEFFFF <1>      call    check_for_backspace
1142 00007D71 7407      <1>      je      short loc_set_time_bs_4
1143                  <1>      ;xor      bh, bh ; video page 0
1144 00007D73 E84DA6FFFF <1>      call    beeper ; BEEP !
1145 00007D78 EBD5      <1>      jmp     short loc_enter_minute_2
1146                  <1> loc_set_time_bs_4:
1147 00007D7A E893FEFFFF <1>      call    write_backspace
1148 00007D7F EB9A      <1>      jmp     short loc_enter_minute_1
1149                  <1>
1150                  <1> set_time_4:
1151                  <1>      ; 13/05/2016
1152                  <1>      ;mov     bx, 7 ; attribute/color (b1)
1153                  <1>      ; video page 0 (bh)
1154 00007D81 B307      <1>      mov     b1, 7
1155 00007D83 E853A5FFFF <1>      call    _write_tty
1156                  <1> loc_enter_time_separator_2:
1157 00007D88 66C705[FA2F0100]30- <1>      mov     word [Second], 3030h
1157 00007D90 30      <1>
1158 00007D91 28E4      <1>      sub     ah, ah
1159 00007D93 E88491FFFF <1>      call    int16h
1160                  <1>      ; AL = ASCII Code of the Character
1161 00007D98 3C0D      <1>      cmp     al, 13
1162                  <1>      ;je      short loc_set_time_progress
1163                  <1>      ; 07/08/2022
1164 00007D9A 7505      <1>      jne     short loc_enter_time_separator_3
1165                  <1> jmp_loc_set_time_progress:
1166 00007D9C E9D4000000 <1>      jmp     loc_set_time_progress
1167                  <1> loc_enter_time_separator_3:
1168 00007DA1 3C1B      <1>      cmp     al, 27
1169                  <1>      ;je      loc_set_time_retn
1170                  <1>      ; 26/07/2022
1171                  <1>      ;je      short loc_set_time_ok
1172                  <1>      ; 07/08/2022
1173 00007DA3 7505      <1>      jne     short loc_enter_time_separator_4
1174 00007DA5 E982000000 <1>      jmp     loc_set_time_ok
1175                  <1> loc_enter_time_separator_4:
1176 00007DAA 3C3A      <1>      cmp     al, ':'
1177 00007DAC 7563      <1>      jne     short loc_set_time_stc_5
1178                  <1>
1179                  <1>      ; 13/05/2016
1180                  <1>      ;mov     bx, 7 ; attribute/color (b1)
1181                  <1>      ; video page 0 (bh)
1182 00007DAE B307      <1>      mov     b1, 7
1183 00007DB0 E826A5FFFF <1>      call    _write_tty
1184                  <1> loc_enter_second_1:
1185 00007DB5 30E4      <1>      xor     ah, ah
1186 00007DB7 E86091FFFF <1>      call    int16h
1187                  <1>      ; AL = ASCII Code of the Character
1188 00007DBC 3C0D      <1>      cmp     al, 13
1189                  <1>      ;je      short loc_set_time_progress
1190                  <1>      ; 07/08/2022
1191 00007DBE 74DC      <1>      je      short jmp_loc_set_time_progress
1192 00007DC0 3C1B      <1>      cmp     al, 27
1193                  <1>      ;je      loc_set_time_retn
1194                  <1>      ; 26/07/2022
1195                  <1>      ;je      short loc_set_time_ok
1196                  <1>      ; 07/08/2022
1197 00007DC2 7502      <1>      jne     short loc_enter_second_2
1198 00007DC4 EB66      <1>      jmp     loc_set_time_ok
1199                  <1> loc_enter_second_2:
1200 00007DC6 A2[FA2F0100] <1>      mov     [Second], al
1201 00007DCB 3C30      <1>      cmp     al, '0'
1202 00007DCD 7267      <1>      jb      short loc_set_time_stc_6
1203 00007DCF 3C35      <1>      cmp     al, '5'
1204 00007DD1 7763      <1>      ja      short loc_set_time_stc_6
1205                  <1>      ; 13/05/2016
1206                  <1>      ;mov     bx, 7 ; attribute/color (b1)
1207                  <1>      ; video page 0 (bh)
1208 00007DD3 B307      <1>      mov     b1, 7
1209 00007DD5 E801A5FFFF <1>      call    _write_tty
1210                  <1> loc_enter_second_3:
1211 00007DDA 30E4      <1>      xor     ah, ah
1212 00007DDC E83B91FFFF <1>      call    int16h
1213                  <1>      ; AL = ASCII Code of the Character
1214 00007DE1 3C1B      <1>      cmp     al, 27
1215                  <1>      ;je      short loc_set_time_retn
1216                  <1>      ; 26/07/2022
1217 00007DE3 7447      <1>      je      short loc_set_time_ok
1218 00007DE5 3C30      <1>      cmp     al, '0'
1219 00007DE7 7271      <1>      jb      short loc_set_time_stc_7
1220 00007DE9 3C39      <1>      cmp     al, '9'
1221 00007DEB 776D      <1>      ja      short loc_set_time_stc_7
1222                  <1>      ; 13/05/2016
1223                  <1>      ;mov     bx, 7 ; attribute/color (b1)
1224                  <1>      ; video page 0 (bh)

```

```

1225 00007DED B307      <1>      mov     bl, 7
1226 00007DEF E8E7A4FFFF <1>      call    _write_tty
1227                      <1> loc_set_time_get_lchar_again:
1228 00007DF4 28E4      <1>      sub     ah, ah ; 0
1229 00007DF6 E82191FFFF <1>      call    int16h
1230                      <1>      ; AL = ASCII Code of the Character
1231 00007DFB 3C0D      <1>      cmp     al, 13
1232 00007DFD 7476      <1>      je      short loc_set_time_progress
1233 00007DFF 3C1B      <1>      cmp     al, 27
1234                      <1>      ;je     short loc_set_time_retn
1235                      <1>      ; 07/08/2022
1236 00007E01 7429      <1>      je      short loc_set_time_ok
1237                      <1>      ;
1238 00007E03 E81CFEFFFF <1>      call    check_for_backspace
1239 00007E08 75EA      <1>      jne     short loc_set_time_get_lchar_again
1240                      <1>
1241                      <1> loc_set_time_bs_8:
1242 00007E0A E803FEFFFF <1>      call    write_backspace
1243 00007E0F EBC9      <1>      jmp     short loc_enter_second_3
1244                      <1>
1245                      <1> ; 26/07/2022
1246                      <1> ;loc_set_time_retn:
1247                      <1> ; mov     esi, nextline
1248                      <1> ; call    print_msg
1249                      <1> ; retn
1250                      <1> ; jmp     print_msg
1251                      <1>
1252                      <1> ; 26/07/2022
1253                      <1> ;loc_set_time_stc_0:
1254                      <1> ; xor     bh, bh ; video page 0
1255                      <1> ; call    beeper ; BEEP !
1256                      <1> ; jmp     loc_enter_hour_1
1257                      <1> ;loc_set_time_stc_1:
1258                      <1> ; call    check_for_backspace
1259                      <1> ; je      short loc_set_time_bs_1
1260                      <1> ; xor     bh, bh ; video page 0
1261                      <1> ; call    beeper ; BEEP !
1262                      <1> ; jmp     loc_enter_hour_2
1263                      <1> ;loc_set_time_bs_1:
1264                      <1> ; call    write_backspace
1265                      <1> ; jmp     loc_enter_hour_1
1266                      <1> ;loc_set_time_stc_2:
1267                      <1> ; call    check_for_backspace
1268                      <1> ; je      short loc_set_time_bs_2
1269                      <1> ; xor     bh, bh ; video page 0
1270                      <1> ; call    beeper ; BEEP !
1271                      <1> ; jmp     loc_enter_time_separator_1
1272                      <1> ;loc_set_time_bs_2:
1273                      <1> ; call    write_backspace
1274                      <1> ; jmp     loc_enter_hour_2
1275                      <1> ;loc_set_time_stc_3:
1276                      <1> ; call    check_for_backspace
1277                      <1> ; je      short loc_set_time_bs_3
1278                      <1> ; xor     bh, bh ; video page 0
1279                      <1> ; call    beeper ; BEEP !6
1280                      <1> ; jmp     loc_enter_minute_1
1281                      <1> ;loc_set_time_bs_3:
1282                      <1> ; call    write_backspace
1283                      <1> ; jmp     loc_enter_time_separator_1
1284                      <1> ;loc_set_time_stc_4:
1285                      <1> ; call    check_for_backspace
1286                      <1> ; je      short loc_set_time_bs_4
1287                      <1> ; xor     bh, bh ; video page 0
1288                      <1> ; call    beeper ; BEEP !
1289                      <1> ; jmp     loc_enter_minute_2
1290                      <1> ;loc_set_time_bs_4:
1291                      <1> ; call    write_backspace
1292                      <1> ; jmp     loc_enter_minute_1
1293                      <1>
1294                      <1> ; 26/07/2022
1295                      <1> ;loc_set_time_stc_5:
1296 00007E11 E80EFEFFFF <1>      call    check_for_backspace
1297 00007E16 740A      <1>      je      short loc_set_time_bs_5
1298                      <1> ; xor     bh, bh ; video page 0
1299 00007E18 E8A8A5FFFF <1>      call    beeper ; BEEP !
1300 00007E1D E966FFFFFF <1>      jmp     loc_enter_time_separator_2
1301                      <1> ;loc_set_time_bs_5:
1302 00007E22 E8EBFDFFFF <1>      call    write_backspace
1303 00007E27 E923FFFFFF <1>      jmp     loc_enter_minute_2
1304                      <1>
1305                      <1> ; 26/07/2022
1306                      <1> ;loc_set_time_ok:
1307 00007E2C BE[07390100] <1>      mov     esi, nextline
1308                      <1> ; call    print_msg
1309                      <1> ; retn
1310 00007E31 E974EFFFFF <1>      jmp     print_msg
1311                      <1>
1312                      <1> ; 07/08/2022
1313                      <1> ;loc_set_time_stc_6:
1314 00007E36 E8E9FDFFFF <1>      call    check_for_backspace
1315 00007E3B 7413      <1>      je      short loc_set_time_bs_6
1316                      <1> ; xor     bh, bh ; video page 0
1317 00007E3D E883A5FFFF <1>      call    beeper ; BEEP !
1318 00007E42 66C705[FA2F0100]30- <1>      mov     word [Second], 3030h
1318 00007E4A 30      <1>
1319 00007E4B E965FFFFFF <1>      jmp     loc_enter_second_1
1320                      <1> ;loc_set_time_bs_6:
1321 00007E50 E8BDFDFFFF <1>      call    write_backspace
1322 00007E55 E92EFFFFFF <1>      jmp     loc_enter_time_separator_2
1323                      <1> ;loc_set_time_stc_7:
1324 00007E5A E8C5FDFFFF <1>      call    check_for_backspace
1325 00007E5F 740A      <1>      je      short loc_set_time_bs_7
1326                      <1> ; xor     bh, bh ; video page 0
1327 00007E61 E85FA5FFFF <1>      call    beeper ; BEEP !
1328 00007E66 E96FFFFFFF <1>      jmp     loc_enter_second_3
1329                      <1> ;loc_set_time_bs_7:
1330 00007E6B E8A2FDFFFF <1>      call    write_backspace
1331 00007E70 E940FFFFFF <1>      jmp     loc_enter_second_1
1332                      <1>
1333                      <1> ;loc_set_time_progress:
1334                      <1> ; Get Current Time
1335                      <1> ; mov     ah, 02h
1336                      <1> ; call    int1Ah
1337 00007E75 E8D7E4FFFF <1>      call    RTC_20 ; GET RTC TIME
1338                      <1> ; DL = Daylight Savings Enable option (0-1)
1339                      <1>
1340 00007E7A 66A1[F42F0100] <1>      mov     ax, [Hour]
1341 00007E80 662D3030      <1>      sub     ax, '00'
1342 00007E84 C0E004      <1>      shl     al, 4 ; * 16
1343 00007E87 88C5      <1>      mov     ch, al
1344 00007E89 00E5      <1>      add     ch, ah
1345 00007E8B 66A1[F72F0100] <1>      mov     ax, [Minute]
1346 00007E91 662D3030      <1>      sub     ax, '00'
1347 00007E95 C0E004      <1>      shl     al, 4 ; * 16

```

```

1348 00007E98 88C1      <1>      mov     cl, al
1349 00007E9A 00E1      <1>      add     cl, ah
1350 00007E9C 66A1[FA2F0100]    <1>      mov     ax, [Second]
1351 00007EA2 662D3030          <1>      sub     ax, '00'
1352 00007EA6 C0E004          <1>      shl     al, 4 ; * 16
1353 00007EA9 88C6          <1>      mov     dh, al
1354 00007EAB 00E6          <1>      add     dh, ah
1355                                <1>
1356                                <1>      ;mov     ah, 03h
1357                                <1>      ;call    int1Ah
1358 00007EAD E8CDE4FFFF    <1>      call    RTC_30 ; SET RTC TIME
1359                                <1>
1360                                <1>      ; 26/07/2022
1361 00007EB2 E975FFFFFF    <1>      jmp     loc_set_time_ok
1362                                <1>
1363                                <1> print_volume_info:
1364                                <1>      ; 19/12/2025 - TRDOS 386 v2.0.10
1365                                <1>      ; 01/03/2016
1366                                <1>      ; 08/02/2016
1367                                <1>      ; 06/02/2016
1368                                <1>      ; 04/02/2016
1369                                <1>      ; 18/01/2016 (TRDOS 386 = TRDOS v2.0)
1370                                <1>      ; 25/10/2009
1371                                <1>
1372                                <1>      ; "Volume Serial No: "
1373                                <1>
1374                                <1>      ; INPUT : AL = DOS Drive Number
1375                                <1>      ; OUTPUT : AH = FS Type
1376                                <1>      ;           AL = DOS Drive Name
1377                                <1>      ; CF = 0 -> OK
1378                                <1>      ; CF = 1 -> Drive not ready
1379                                <1>
1380 00007EB7 88C4          <1>      mov     ah, al
1381 00007EB9 28C0          <1>      sub     al, al
1382 00007EBB 0FB7F0          <1>      movzx   esi, ax
1383 00007EBE 81C600010900    <1>      add     esi, Logical_DOSDisks
1384 00007EC4 8A06          <1>      mov     al, [esi]
1385 00007EC6 3C41          <1>      cmp     al, 'A'
1386 00007EC8 7304          <1>      jnb     short loc_pvi_set_vol_name
1387 00007ECA 8A6604          <1>      mov     ah, [esi+LD_FSType]
1388 00007ECD C3          <1>      retn
1389                                <1>
1390                                <1> loc_pvi_set_vol_name:
1391 00007ECE A2[2E300100]    <1>      mov     [Vol_Drv_Name], al
1392 00007ED3 56          <1>      push    esi
1393 00007ED4 E833010000    <1>      call    move_volume_name_and_serial_no ;;;
1394 00007ED9 7302          <1>      jnc     short loc_pvi_mvn_ok
1395 00007EDB 5E          <1>      pop     esi
1396 00007EDC C3          <1>      retn
1397                                <1>
1398                                <1> loc_pvi_mvn_ok:
1399 00007EDD 8B3424          <1>      mov     esi, [esp]
1400                                <1>
1401                                <1>      ; 19/12/2025
1402                                <1> %if 0
1403                                <1>      cmp     byte [esi+LD_FSType], 0A1h
1404                                <1>      jne     short loc_pvi_fat_vol_size
1405                                <1>      mov     eax, [esi+LD_FS_VolumeSize]
1406                                <1>      ; 19/12/2025
1407                                <1>      ;movzx   ebx, word [esi+LD_FS_BytesPerSec]
1408                                <1>      jmp     short loc_vol_size_mul32
1409                                <1> %endif
1410                                <1>
1411                                <1> loc_pvi_fat_vol_size:
1412 00007EE0 8B4670          <1>      mov     eax, [esi+LD_TotalSectors]
1413 00007EE3 8B4670          <1>      mov     eax, [esi+LD_TotalSectors]
1414                                <1>      ; 19/12/2025
1415                                <1>      ;movzx   ebx, word [esi+LD_BPB+BPB_BytsPerSec]
1416                                <1> loc_vol_size_mul32:
1417                                <1>      ; 19/12/2025
1418 00007EE6 BB00020000    <1>      mov     ebx, 512
1419 00007EEB F7E3          <1>      mul     ebx
1420 00007EED 09D2          <1>      or      edx, edx
1421 00007EEF 7507          <1>      jnz     short loc_vol_size_in_kbytes
1422                                <1> loc_vol_size_in_bytes:
1423 00007EF1 B9[0C300100]    <1>      mov     ecx, VolSize_Bytes
1424 00007EF6 EB0B          <1>      jmp     short loc_write_vol_size_str
1425                                <1> loc_vol_size_in_kbytes:
1426                                <1>      ;mov     bx, 1024
1427                                <1>      ; 19/12/2025
1428 00007EF8 D1E3          <1>      shl     ebx, 1 ; ebx = 1024
1429 00007EFA F7F3          <1>      div     ebx
1430 00007EFC B9[FF2F0100]    <1>      mov     ecx, VolSize_KiloBytes
1431 00007F01 31D2          <1>      xor     edx, edx ; 0
1432                                <1> loc_write_vol_size_str:
1433 00007F03 890D[987E0100]    <1>      mov     [VolSize_Unit1], ecx
1434                                <1>      ;
1435 00007F09 BF[AE7E0100]    <1>      mov     edi, Vol_Tot_Sec_Str_End
1436                                <1>      ;mov     byte [edi], 0
1437 00007F0E B90A000000    <1>      mov     ecx, 10
1438                                <1> loc_write_vol_size_chr:
1439 00007F13 F7F1          <1>      div     ecx
1440 00007F15 80C230          <1>      add     dl, '0'
1441 00007F18 4F          <1>      dec     edi
1442 00007F19 8817          <1>      mov     [edi], dl
1443 00007F1B 85C0          <1>      test    eax, eax
1444 00007F1D 7404          <1>      jz      short loc_write_vol_size_str_ok
1445 00007F1F 28D2          <1>      sub     dl, dl ; 0
1446 00007F21 EBF0          <1>      jmp     short loc_write_vol_size_chr
1447                                <1>
1448                                <1> loc_write_vol_size_str_ok:
1449 00007F23 893D[A07E0100]    <1>      mov     [Vol_Tot_Sec_Str_Start], edi
1450                                <1>      ;
1451 00007F29 BF[17300100]    <1>      mov     edi, Vol_FS_Name
1452 00007F2E 8A4E03          <1>      mov     cl, [esi+LD_FATType]
1453                                <1>
1454                                <1>      ; 19/12/2025
1455                                <1> %if 0
1456                                <1>      and     cl, cl ; 0 ?
1457                                <1>      jnz     short loc_write_vol_FAT_str_1
1458                                <1>      mov     word [edi], 'TR'
1459                                <1>      mov     dword [edi+4], ' FS1'
1460                                <1>      ; 19/12/2025
1461                                <1>      ;movzx   ebx, word [esi+LD_FS_BytesPerSec]
1462                                <1>      ;mov     bx, [esi+LD_FS_BytesPerSec]
1463                                <1>      mov     eax, [esi+LD_FS_FreeSectors]
1464                                <1>      jmp     short loc_vol_freespace_mul32
1465                                <1> %endif
1466                                <1>
1467                                <1> loc_write_vol_FAT_str_1:
1468 00007F31 66B83332          <1>      mov     ax, '32' ; FAT32
1469 00007F35 80F902          <1>      cmp     cl, 2 ; [esi+LD_FATType]
1470 00007F38 7708          <1>      ja     short loc_write_vol_FAT_str_2
1471 00007F3A 66B83132          <1>      mov     ax, '12' ; FAT12

```

```

1472 00007F3E 7202      <1>      jb      short loc_write_vol_FAT_str_2
1473 00007F40 B436      <1>      mov     ah, '6' ; FAT16
1474                                <1> loc_write_vol_FAT_str_2:
1475 00007F42 C70746415420 <1>      mov     dword [edi], 'FAT '
1476 00007F48 66894704   <1>      mov     word [edi+4], ax
1477                                <1>      ; 19/12/2025
1478                                <1>      ;;movzx ebx, word [esi+LD_BPB+BPB_BytsPerSec]
1479                                <1>      ;mov     bx, [esi+LD_BPB+BPB_BytsPerSec]
1480 00007F4C 8B4674     <1>      mov     eax, [esi+LD_FreeSectors]
1481                                <1>
1482                                <1> loc_vol_freespace_recalc0:
1483                                <1>      ; 01/03/2016
1484 00007F4F 83F8FF     <1>      cmp     eax, 0FFFFFFFh
1485 00007F52 7209     <1>      jb      short loc_vol_freespace_mul32
1486                                <1>      ;inc     eax ; 0
1487                                <1>
1488                                <1> ; 19/12/2025
1489                                <1> %if 0
1490                                <1>      and     cl, cl ; byte [esi+LD_FATType]
1491                                <1>      jz      short loc_vol_freespace_mul32
1492                                <1> %endif
1493                                <1>      ; 19/12/2025
1494                                <1>      ;push     ebx
1495 00007F54 66BB00FF     <1>      mov     bx, 0FF00h ; recalculate free sectors
1496 00007F58 E804460000 <1>      call    calculate_fat_freespace
1497                                <1>      ;pop     ebx
1498                                <1>
1499                                <1>      ; 19/12/2025
1500                                <1>      ; (if cf = 1, eax = -1)
1501                                <1>      ; if eax = -1 , free sector count is invalid
1502                                <1>
1503                                <1> loc_vol_freespace_mul32:
1504 00007F5D B9[0C300100] <1>      mov     ecx, VolSize_Bytes
1505 00007F62 BF[BE7E0100] <1>      mov     edi, Vol_Free_Sectors_Str_End
1506                                <1>      ;mov     byte [edi], 0
1507 00007F67 40         <1>      inc     eax ; * ; -1 -> 0
1508 00007F68 7506     <1>      jnz     short loc_vol_freespace_mul32_@
1509 00007F6A 4F         <1>      dec     edi
1510 00007F6B C6073F     <1>      mov     byte [edi], '?'
1511 00007F6E EB32     <1>      jmp     short loc_write_vol_fspace_str_ok
1512                                <1>
1513                                <1> loc_vol_freespace_mul32_@:
1514 00007F70 48         <1>      dec     eax ; *
1515                                <1>      ; 19/12/2025
1516 00007F71 BB00020000 <1>      mov     ebx, 512
1517 00007F76 F7E3     <1>      mul     ebx
1518 00007F78 09D2     <1>      or      edx, edx
1519                                <1>      ;jnz     short loc_vol_fspace_in_kbytes
1520                                <1> loc_vol_fspace_in_bytes:
1521                                <1>      ; 19/12/2025
1522                                <1>      ;mov     ecx, VolSize_Bytes
1523                                <1>      ;jmp     short loc_write_vol_fspace_str
1524 00007F7A 740B     <1>      jz      short loc_write_vol_fspace_str
1525                                <1> loc_vol_fspace_in_kbytes:
1526                                <1>      ;mov     bx, 1024
1527                                <1>      ; 19/12/2025
1528 00007F7C D1E3     <1>      shl     ebx, 1 ; ebx = 1024
1529 00007F7E F7F3     <1>      div     ebx
1530 00007F80 B9[FF2F0100] <1>      mov     ecx, VolSize_KiloBytes
1531 00007F85 31D2     <1>      xor     edx, edx ; 0
1532                                <1> loc_write_vol_fspace_str:
1533 00007F87 890D[9C7E0100] <1>      mov     [VolSize_Unit2], ecx
1534                                <1>      ;
1535                                <1>      ; 19/12/2025
1536                                <1>      ;mov     edi, Vol_Free_Sectors_Str_End
1537                                <1>      ;mov     byte [edi], 0
1538 00007F8D B90A000000 <1>      mov     ecx, 10
1539                                <1> loc_write_vol_fspace_chr:
1540 00007F92 F7F1     <1>      div     ecx
1541 00007F94 80C230     <1>      add     dl, '0'
1542 00007F97 4F         <1>      dec     edi
1543 00007F98 8817     <1>      mov     [edi], dl
1544 00007F9A 85C0     <1>      test    eax, eax
1545 00007F9C 7404     <1>      jz      short loc_write_vol_fspace_str_ok
1546 00007F9E 28D2     <1>      sub     dl, dl ; 0
1547 00007FA0 EBF0     <1>      jmp     short loc_write_vol_fspace_chr
1548                                <1>
1549                                <1> loc_write_vol_fspace_str_ok:
1550 00007FA2 893D[B07E0100] <1>      mov     [Vol_Free_Sectors_Str_Start], edi
1551                                <1>      ;
1552 00007FA8 BE[15300100] <1>      mov     esi, Volume_in_drive
1553 00007FAD E8F8EDFFFF <1>      call    print_msg
1554 00007FB2 BE[55300100] <1>      mov     esi, Vol_Name
1555 00007FB7 E8EEEDFFFF <1>      call    print_msg
1556 00007FBC BE[07390100] <1>      mov     esi, nextline
1557 00007FC1 E8E4EDFFFF <1>      call    print_msg
1558                                <1>      ;
1559 00007FC6 BE[B6300100] <1>      mov     esi, Vol_Total_Sector_Header
1560 00007FCB E8DAEDFFFF <1>      call    print_msg
1561 00007FD0 8B35[A07E0100] <1>      mov     esi, [Vol_Tot_Sec_Str_Start]
1562 00007FD6 E8CFEDFFFF <1>      call    print_msg
1563 00007FDB 8B35[987E0100] <1>      mov     esi, [VolSize_Unit1]
1564 00007FE1 E8C4EDFFFF <1>      call    print_msg
1565                                <1>      ;
1566 00007FE6 BE[C7300100] <1>      mov     esi, Vol_Free_Sectors_Header
1567 00007FEB E8BAEDFFFF <1>      call    print_msg
1568 00007FF0 8B35[B07E0100] <1>      mov     esi, [Vol_Free_Sectors_Str_Start]
1569 00007FF6 E8AFEDFFFF <1>      call    print_msg
1570 00007FFB 8B35[9C7E0100] <1>      mov     esi, [VolSize_Unit2]
1571 00008001 E8A4EDFFFF <1>      call    print_msg
1572                                <1>      ;
1573 00008006 5E         <1>      pop     esi
1574                                <1>
1575                                <1>      ;mov     ah, [esi+LD_FSType]
1576                                <1>      ;mov     al, [esi+LD_FATType]
1577 00008007 668B4603 <1>      mov     ax, [esi+LD_FATType]
1578                                <1>
1579 0000800B C3         <1>      retn
1580                                <1>
1581                                <1> move_volume_name_and_serial_no:
1582                                <1>      ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
1583                                <1>      ; 26/07/2022 (TRDOS 386 kernel v2.0.5)
1584                                <1>      ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
1585                                <1>      ; this routine will be called by
1586                                <1>      ; "print_volume_info" and "print_directory"
1587                                <1>      ; INPUT ->
1588                                <1>      ; ESI = Logical DOS drv descripton table address
1589                                <1>      ; OUTPUT ->
1590                                <1>      ; *volume name will be moved to text area
1591                                <1>      ; *volume serial number will be converted to
1592                                <1>      ; text and will be moved to text area
1593                                <1>      ; cf = 1 -> invalid/unknown dos drive
1594                                <1>      ; cf = 0 -> ecx = 0
1595                                <1>      ;

```

```

1596      <1>      ; (eax, edx, ecx, esi, edi will be changed)
1597      <1>
1598      <1>      ; 26/07/2022
1599 0000800C 31C9      <1>      xor     ecx, ecx
1600      <1>
1601 0000800E BF[55300100] <1>      mov     edi, vol_Name
1602      <1>
1603      <1>      ;mov     ah, [esi+LD_FSType]
1604      <1>      ; 19/12/2025
1605 00008013 8A4603      <1>      mov     al, [esi+LD_FATType]
1606      <1>      ;mov     ax, [esi+LD_FATType]
1607      <1>
1608      <1>      ; 19/12/2025
1609      <1>      %if 0
1610      <1>      cmp     ah, 0A1h
1611      <1>      je      short mvn_2
1612      <1>      or      ah, ah
1613      <1>      jz      short mvn_0
1614      <1>      or      al, al
1615      <1>      jnz     short mvn_1
1616      <1>      mvn_0:
1617      <1>      mov     al, [esi]
1618      <1>      stc
1619      <1>      retn
1620      <1>      %endif
1621      <1>
1622      <1>      mvn_1:
1623 00008016 3C02      <1>      cmp     al, 2
1624 00008018 7708      <1>      ja      short mvn_3
1625      <1>      ;or      al, al
1626      <1>      ;jz      short mvn_2
1627 0000801A 8B462D      <1>      mov     eax, [esi+LD_BPB+VolumeID]
1628 0000801D 83C631      <1>      add     esi, LD_BPB+VolumeLabel
1629 00008020 EB06      <1>      jmp     short mvn_4
1630      <1>
1631      <1>      ; 19/12/2025
1632      <1>      %if 0
1633      <1>      mvn_2:
1634      <1>      mov     eax, [esi+LD_FS_VolumeSerial]
1635      <1>      add     esi, LD_FS_VolumeName
1636      <1>      ;mov     ecx, 16
1637      <1>      ; 26/07/2022
1638      <1>      mov     cl, 16
1639      <1>      rep     movsd
1640      <1>      jmp     short mvn_5
1641      <1>      %endif
1642      <1>
1643      <1>      mvn_3:
1644 00008022 8B4649      <1>      mov     eax, [esi+LD_BPB+FAT32_VolID]
1645 00008025 83C64D      <1>      add     esi, LD_BPB+FAT32_VolLab
1646      <1>      mvn_4:
1647      <1>      ;mov     ecx, 11
1648      <1>      ; 26/07/2022
1649 00008028 B10B      <1>      mov     cl, 11
1650 0000802A F3A4      <1>      rep     movsb
1651 0000802C C60700      <1>      mov     byte [edi], 0
1652      <1>      mvn_5:
1653      <1>      ;mov     [Current_VolSerial], eax
1654 0000802F E8C7C1FFFF      <1>      call    dwordtohex
1655 00008034 8915[AA300100] <1>      mov     [Vol_Serial1], edx
1656 0000803A A3[AF300100] <1>      mov     [Vol_Serial2], eax
1657      <1>      ; ecx = 0
1658 0000803F C3      <1>      retn
1659      <1>
1660      <1>      get_volume_serial_number:
1661      <1>      ; 19/12/2025 (TRDOS 386 Kernel v2.0.10)
1662      <1>      ; 19/01/2016 (TRDOS 386 = TRDOS v2.0)
1663      <1>      ; 08/08/2010
1664      <1>      ;
1665      <1>      ; INPUT -> DL = Logical DOS Drive number
1666      <1>      ; OUTPUT -> EAX = Volume serial number
1667      <1>      ;
1668      <1>      ; BL= FAT Type
1669      <1>      ; BH = Logical DOS drv Number (DL input)
1670      <1>      ; cf = 1 -> Drive not ready
1671 00008040 31DB      <1>      xor     ebx, ebx
1672 00008042 88D7      <1>      mov     bh, dl
1673 00008044 3815[7D2E0100] <1>      cmp     [Last_DOS_DiskNo], dl
1674 0000804A 7304      <1>      jnb     short loc_gvsn_start
1675      <1>      loc_gvsn_stc_retn:
1676 0000804C 31C0      <1>      xor     eax, eax
1677 0000804E F9      <1>      stc
1678 0000804F C3      <1>      retn
1679      <1>      loc_gvsn_start:
1680 00008050 56      <1>      push    esi
1681 00008051 BE00010900 <1>      mov     esi, Logical_DOSDisks
1682 00008056 01DE      <1>      add     esi, ebx
1683      <1>      ; 19/12/2025
1684      <1>      %if 0
1685      <1>      mov     bl, [esi+LD_FATType]
1686      <1>      and     bl, bl
1687      <1>      jz      short loc_gvsn_fs
1688      <1>      %endif
1689 00008058 80FB02      <1>      cmp     bl, 2
1690 0000805B 7705      <1>      ja      short loc_gvsn_fat32
1691      <1>      loc_gvsn_fat:
1692      <1>      add     esi, LD_BPB + VolumeID
1693 00008060 EB03      <1>      jmp     short loc_gvsn_return
1694      <1>      loc_gvsn_fat32:
1695 00008062 83C649      <1>      add     esi, LD_BPB + FAT32_VolID
1696      <1>      ; 19/12/2025
1697      <1>      %if 0
1698      <1>      jmp     short loc_gvsn_return
1699      <1>      loc_gvsn_fs:
1700      <1>      cmp     byte [esi+LD_FSType], 0A1h
1701      <1>      jne     short loc_gvsn_stc_retn
1702      <1>      add     esi, LD_FS_VolumeSerial
1703      <1>      %endif
1704      <1>
1705      <1>      loc_gvsn_return:
1706 00008065 8B06      <1>      mov     eax, [esi]
1707 00008067 5E      <1>      pop     esi
1708 00008068 C3      <1>      retn
1709      <1>
1710      <1>      ; CMD_INTR.ASM [ TRDOS Command Interpreter Procedure ]
1711      <1>      ; 09/11/2011
1712      <1>      ; 29/01/2005
1713      <1>
1714      <1>      command_interpreter:
1715      <1>      ; 26/07/2022 (TRDOS 386 Kernel v2.0.5)
1716      <1>      ; 16/10/2016
1717      <1>      ; 12/10/2016
1718      <1>      ; 13/05/2016
1719      <1>      ; 07/05/2016

```



```

1720      <1>      ; 04/03/2016
1721      <1>      ; 04/02/2016
1722      <1>      ; 03/02/2016
1723      <1>      ; 30/01/2016
1724      <1>      ; 29/01/2016 (TRDOS 386 = TRDOS 2.0)
1725      <1>      ; 15/09/2011
1726      <1>      ; 29/01/2005
1727      <1>
1728      <1>      ; Input: ecx = command word length (CL)
1729      <1>      ;      CommandBuffer = Command string offset
1730      <1>
1731      00008069 C605[507F0100]00      <1>      mov     byte [Program_Exit], 0
1732      <1>
1733      <1>      ;cmp     cl, 4
1734      <1>      ;ja      c_6
1735      <1>      ;jb      c_2
1736      <1>
1737      <1>      ; 26/07/2022
1738      00008070 80F903      <1>      cmp     cl, 3
1739      00008073 777C      <1>      ja      short c_4
1740      00008075 7405      <1>      je      short c_3
1741      00008077 E9D2010000      <1>      jmp     c_2
1742      <1>      c_3:
1743      <1>      cmp_cmd_dir:
1744      0000807C BFE62E0100]      <1>      mov     edi, Cmd_Dir
1745      00008081 E8D1030000      <1>      call    cmp_cmd
1746      <1>      ;jnc     print_directory_list
1747      <1>      ; 26/07/2022
1748      00008086 7205      <1>      jc      short cmp_cmd_cls
1749      00008088 E97D040000      <1>      jmp     print_directory_list
1750      <1>
1751      <1>      cmp_cmd_cls:
1752      0000808D B103      <1>      mov     cl, 3
1753      0000808F BFE222F0100]      <1>      mov     edi, Cmd_Cls
1754      00008094 E8BE030000      <1>      call    cmp_cmd
1755      <1>      ;jnc     clear_screen
1756      <1>      ; 26/07/2022
1757      00008099 7205      <1>      jc      short cmp_cmd_ver
1758      0000809B E920EDFFFF      <1>      jmp     clear_screen
1759      <1>
1760      <1>      cmp_cmd_ver:
1761      000080A0 B103      <1>      mov     cl, 3
1762      000080A2 BFEF02E0100]      <1>      mov     edi, Cmd_Ver
1763      000080A7 E8AB030000      <1>      call    cmp_cmd
1764      000080AC 720A      <1>      jc      short cmp_cmd_mem
1765      <1>
1766      000080AE BE[852E0100]      <1>      mov     esi, mainprog_Version
1767      <1>      ;call    print_msg
1768      000080B3 E9F2ECFFFF      <1>      jmp     print_msg
1769      <1>      ;retn
1770      <1>
1771      <1>      cmp_cmd_mem:
1772      000080B8 B103      <1>      mov     cl, 3
1773      000080BA BFE582F0100]      <1>      mov     edi, Cmd_Mem
1774      000080BF E893030000      <1>      call    cmp_cmd
1775      <1>      ;jnc     memory_info
1776      <1>      ; 26/07/2022
1777      000080C4 7205      <1>      jc      short cmp_cmd_del
1778      000080C6 E95DC0FFFF      <1>      jmp     memory_info
1779      <1>
1780      <1>      cmp_cmd_del:
1781      000080CB B103      <1>      mov     cl, 3
1782      000080CD BFE2B2F0100]      <1>      mov     edi, Cmd_Del
1783      000080D2 E880030000      <1>      call    cmp_cmd
1784      <1>      ;jnc     delete_file
1785      <1>      ; 26/07/2022
1786      000080D7 7205      <1>      jc      short cmp_cmd_set
1787      000080D9 E9E2100000      <1>      jmp     delete_file
1788      <1>
1789      <1>      cmp_cmd_set:
1790      000080DE B103      <1>      mov     cl, 3
1791      000080E0 BFE1E2F0100]      <1>      mov     edi, Cmd_Set
1792      000080E5 E86D030000      <1>      call    cmp_cmd
1793      <1>      ;jnc     set_get_env
1794      <1>      ; 26/07/2022
1795      000080EA 720F      <1>      jc      short cmp_cmd_run
1796      000080EC E9F1180000      <1>      jmp     set_get_env
1797      <1>
1798      <1>      ; 07/08/2022
1799      <1>      c_4:
1800      <1>      ; 26/07/2022
1801      000080F1 80F904      <1>      cmp     cl, 4
1802      000080F4 741D      <1>      je      short cmp_cmd_4
1803      000080F6 E937020000      <1>      jmp     c_6
1804      <1>
1805      <1>      cmp_cmd_run:
1806      000080FB B103      <1>      mov     cl, 3
1807      000080FD BFE1A2F0100]      <1>      mov     edi, Cmd_Run
1808      00008102 E850030000      <1>      call    cmp_cmd
1809      <1>      ; 07/05/2016
1810      <1>      ;jc     cmp_cmd_external
1811      <1>      ; 26/07/2022
1812      00008107 7305      <1>      jnc     short c3_run
1813      00008109 E92C030000      <1>      jmp     cmp_cmd_external
1814      <1>      c3_run:
1815      0000810E E9D71E0000      <1>      jmp     load_and_execute_file
1816      <1>
1817      <1>      cmp_cmd_4:
1818      <1>      ; 26/07/2022
1819      <1>      cmp_cmd_exit:
1820      00008113 BFEF42E0100]      <1>      mov     edi, Cmd_Exit
1821      00008118 E83A030000      <1>      call    cmp_cmd
1822      0000811D 7208      <1>      jc      short cmp_cmd_date
1823      <1>
1824      0000811F C605[507F0100]01      <1>      mov     byte [Program_Exit], 1
1825      00008126 C3      <1>      retn
1826      <1>
1827      <1>      cmp_cmd_date:
1828      00008127 B104      <1>      mov     cl, 4
1829      00008129 BFE102F0100]      <1>      mov     edi, Cmd_Date
1830      0000812E E824030000      <1>      call    cmp_cmd
1831      00008133 720A      <1>      jc      short cmp_cmd_time
1832      <1>
1833      00008135 E85BF8FFFF      <1>      call    show_date
1834      <1>      ;call    set_date
1835      <1>      ;retn
1836      <1>      ; 26/07/2022
1837      0000813A E999F8FFFF      <1>      jmp     set_date
1838      <1>
1839      <1>      cmp_cmd_time:
1840      0000813F B104      <1>      mov     cl, 4
1841      00008141 BFE152F0100]      <1>      mov     edi, Cmd_Time
1842      00008146 E80C030000      <1>      call    cmp_cmd
1843      0000814B 720A      <1>      jc      short cmp_cmd_show

```

```

1844
1845 0000814D E8E9FAFFFF <1> call show_time
1846 <1> ;call set_time
1847 <1> ;retn
1848 <1> ; 26/07/2022
1849 00008152 E91AFBFFFF <1> jmp set_time
1850 <1>
1851 <1> cmp_cmd_show:
1852 00008157 B104 <1> mov cl, 4
1853 00008159 BF[262F0100] <1> mov edi, Cmd_Show
1854 0000815E E8F4020000 <1> call cmp_cmd
1855 <1> ;jnc show_file
1856 <1> ; 26/07/2022
1857 00008163 7205 <1> jc short cmp_cmd_echo
1858 00008165 E971090000 <1> jmp show_file
1859 <1>
1860 <1> cmp_cmd_echo:
1861 0000816A B104 <1> mov cl, 4
1862 0000816C BF[622F0100] <1> mov edi, Cmd_Echo
1863 00008171 E8E1020000 <1> call cmp_cmd
1864 00008176 7224 <1> jc short cmp_cmd_copy
1865 <1>
1866 <1> ; 22/11/2017
1867 <1> ; AL = 0
1868 00008178 803E20 <1> cmp byte [esi], 20h
1869 0000817B 7215 <1> jnb short cmd_echo_nextline
1870 <1> ; 14/04/2016
1871 0000817D 56 <1> push esi
1872 <1> cmd_echo_asciiz:
1873 <1> ;inc esi
1874 <1> ;mov al, [esi]
1875 <1> ; 22/11/2017
1876 0000817E AC <1> lodsb
1877 0000817F 3C20 <1> cmp al, 20h
1878 00008181 73FB <1> jnb short cmd_echo_asciiz
1879 00008183 4E <1> dec esi
1880 00008184 C60600 <1> mov byte [esi], 0
1881 00008187 5E <1> pop esi
1882 00008188 89F7 <1> mov edi, esi
1883 0000818A E81BECFFFF <1> call print_msg
1884 0000818F C60700 <1> mov byte [edi], 0
1885 <1> cmd_echo_nextline:
1886 00008192 BE[75390100] <1> mov esi, NextLine
1887 <1> ;call print_msg
1888 <1> ;retn
1889 00008197 E90EECFFFF <1> jmp print_msg
1890 <1>
1891 <1> cmp_cmd_copy:
1892 0000819C B104 <1> mov cl, 4
1893 0000819E BF[492F0100] <1> mov edi, Cmd_Copy
1894 000081A3 E8AF020000 <1> call cmp_cmd
1895 <1> ;jnc copy_file
1896 <1> ; 26/07/2022
1897 000081A8 7205 <1> jc short cmp_cmd_move
1898 000081AA E995160000 <1> jmp copy_file
1899 <1>
1900 <1> cmp_cmd_move:
1901 000081AF B104 <1> mov cl, 4
1902 000081B1 BF[4E2F0100] <1> mov edi, Cmd_Move
1903 000081B6 E89C020000 <1> call cmp_cmd
1904 <1> ;jnc move_file
1905 <1> ; 26/07/2022
1906 000081BB 7205 <1> jc short cmp_cmd_path
1907 000081BD E940150000 <1> jmp move_file
1908 <1>
1909 <1> cmp_cmd_path:
1910 000081C2 B104 <1> mov cl, 4
1911 000081C4 BF[532F0100] <1> mov edi, Cmd_Path
1912 000081C9 E889020000 <1> call cmp_cmd
1913 <1> ;jnc set_get_path
1914 <1> ; 26/07/2022
1915 000081CE 7205 <1> jc short cmp_cmd_beep
1916 000081D0 E995180000 <1> jmp set_get_path
1917 <1>
1918 <1> cmp_cmd_beep:
1919 000081D5 B104 <1> mov cl, 4
1920 000081D7 BF[802F0100] <1> mov edi, Cmd_Beep
1921 000081DC E876020000 <1> call cmp_cmd
1922 000081E1 720B <1> jc short cmp_cmd_find
1923 <1> ; 13/05/2016
1924 000081E3 8A3D[CE760100] <1> mov bh, [ptty] ; [ACTIVE_PAGE]
1925 000081E9 E9D7A1FFFF <1> jmp beeper
1926 <1>
1927 <1> cmp_cmd_find:
1928 000081EE B104 <1> mov cl, 4
1929 000081F0 BF[5D2F0100] <1> mov edi, Cmd_Find
1930 000081F5 E85D020000 <1> call cmp_cmd
1931 <1> ;jc cmp_cmd_external
1932 <1> ; 26/07/2022
1933 000081FA 7305 <1> jnc short c4_find
1934 000081FC E939020000 <1> jmp cmp_cmd_external
1935 <1> c4_find:
1936 <1> ;call find_and_list_files
1937 00008201 E9FC200000 <1> jmp find_and_list_files
1938 <1> ;retn
1939 <1>
1940 <1> c_1:
1941 00008206 AD <1> lodsd
1942 <1> cmp_cmd_help:
1943 00008207 3C3F <1> cmp al, '?'
1944 00008209 751D <1> jne short cmp_cmd_remark
1945 <1>
1946 0000820B BE[E62E0100] <1> mov esi, Command_List
1947 <1> cmd_help_next_w:
1948 00008210 E895EBFFFF <1> call print_msg
1949 <1>
1950 00008215 803E20 <1> cmp byte [esi], 20h ; 0
1951 00008218 7233 <1> jnb short cmd_help_retn
1952 <1>
1953 0000821A 56 <1> push esi
1954 0000821B BE[07390100] <1> mov esi, nextline
1955 00008220 E885EBFFFF <1> call print_msg
1956 00008225 5E <1> pop esi
1957 00008226 EBE8 <1> jmp short cmd_help_next_w
1958 <1>
1959 <1> cmp_cmd_remark:
1960 00008228 3C2A <1> cmp al, '*'
1961 <1> ;jne cmp_cmd_external
1962 <1> ; 26/07/2022
1963 0000822A 7405 <1> je short cmp_cmd_rem
1964 0000822C E909020000 <1> jmp cmp_cmd_external
1965 <1> cmp_cmd_rem:
1966 00008231 46 <1> inc esi
1967 00008232 BF[C4770100] <1> mov edi, Remark

```

```

1968 00008237 8A06      <1>      mov     al, [esi]
1969 00008239 3C20      <1>      cmp     al, 20h
1970 0000823B 7707      <1>      ja      short cmd_remark_write
1971 0000823D 89FE      <1>      mov     esi, edi ; Remark
1972 0000823F E966EBFFFF <1>      jmp     print_msg
1973
1974
1975 00008244 AA        <1>      cmd_remark_write:
1976 00008245 AC        <1>      stosb
1977 00008246 3C20      <1>      lodsb
1978 00008248 73FA      <1>      cmp     al, 20h
1979 0000824A C60700    <1>      jnb     short cmd_remark_write
1980
1981
1982
1983
1984 0000824D C3        <1>      mov     byte [edi], 0
1985
1986
1987 0000824E BE[12780100] <1>      cmd_help_retn:
1988 00008253 80F902    <1>      cmd_remark_retn:
1989
1990
1991 00008256 72AE      <1>      cd_retn:
1992
1993
1994
1995
1996 00008258 66AD      <1>      retn
1997 0000825A 663D4344  <1>      c_2:
1998 0000825E 754E      <1>      ; 26/07/2022
1999 00008260 46        <1>      mov     esi, CommandBuffer
2000
2001 00008261 668B06    <1>      cmp     cl, 2
2002 00008264 3C20      <1>      ;ja     c_3
2003 00008266 76E5      <1>      ;mov    esi, CommandBuffer
2004
2005 00008268 80FC3A    <1>      jnb     short c_1
2006 0000826B 7504      <1>      ; 26/07/2022
2007 0000826D 46        <1>      ;jne    short c_1
2008 0000826E 46        <1>
2009 0000826F EB47      <1>      cmp_cmd_cd:
2010
2011
2012
2013
2014
2015
2016
2017
2018 00008271 B4CD      <1>      lodsw
2019
2020 00008273 E863210000 <1>      cmp     ax, 'CD'
2021
2022
2023 00008278 7205      <1>      jne     short cmp_cmd_drive
2024 0000827A E985200000 <1>      inc     esi
2025
2026
2027 0000827F 3C03      <1>      cd_0:
2028 00008281 740C      <1>      mov     ax, [esi]
2029
2030 00008283 3C0F      <1>      cmp     al, 20h
2031 00008285 7453      <1>      jna     short cd_retn
2032 00008287 3C11      <1>      ; 10/02/2016
2033 00008289 744F      <1>      cmp     ah, ':'
2034 0000828B 3C13      <1>      jne     short cd_1
2035 0000828D 7460      <1>      inc     esi
2036
2037
2038 0000828F 50        <1>      inc     esi
2039
2040 00008290 BE[89310100] <1>      jmp     short cd_2
2041 00008295 E810EBFFFF <1>
2042
2043 0000829A 58        <1>      cd_1: ; change current directory
2044 0000829B 3A25[60770100] <1>      ; 29/11/2009
2045
2046
2047 000082A1 7306      <1>      ; AH = CDh ; to separate 'CD' command from others
2048 000082A3 8825[60770100] <1>      ; for restoring current directory
2049
2050 000082A9 E956200000 <1>      ; 0CDh sign is for saving cdir into
2051
2052
2053
2054 000082AE 80FC3A    <1>      ; DOS drv description table cdir area
2055
2056
2057 000082B1 7405      <1>      mov     ah, 0CDh ; mov byte [CD_COMMAND], 0CDh
2058
2059 000082B3 E982010000 <1>
2060
2061
2062 000082B8 803E20    <1>      call    change_current_directory
2063
2064
2065 000082BB 7706      <1>      ;jnc    change_prompt_dir_string
2066 000082BD 24DF      <1>      ; 26/07/2022
2067 000082BF 2C41      <1>      ;jc     short cd_error_messages
2068
2069
2070 000082C1 7305      <1>      jmp     change_prompt_dir_string
2071
2072 000082C3 E97C010000 <1>
2073
2074 000082C8 3A05[7D2E0100] <1>      cd_error_messages:
2075 000082CE 770A      <1>      cmp     al, 3
2076
2077 000082D0 88C2      <1>      je      short cd_path_not_found
2078 000082D2 E809F4FFFF <1>      ; 16/10/2016 (15h -> 15)
2079 000082D7 7201      <1>      cmp     al, 15 ; drive not ready error
2080 000082D9 C3        <1>      je      short cd_drive_not_ready
2081
2082
2083 000082DA BE[46310100] <1>      cmp     al, 17 ; read error
2084 000082DF E8C6EAFFFF <1>      je      short cd_drive_not_ready
2085
2086
2087 000082E4 8A15[62770100] <1>      cmp     al, 19 ; ; Bad directory/path name
2088
2089 000082EA E9F1F3FFFF <1>      je      short cd_command_failed
2090
2091

```

```

2092                                     <1> cd_command_failed:
2093 000082EF BE[27310100]             <1>     mov     esi, Msg_Bad_Command
2094 000082F4 E8B1EAF0FF             <1>     call    print_msg
2095 000082F9 EBE9                   <1>     jmp     short cd_fail_drive_restart
2096                                     <1>
2097                                     <1> c_5:
2098                                     <1> cmp_cmd_mkdir:
2099 000082FB BF[432F0100]             <1>     mov     edi, Cmd_Mkdir
2100 00008300 E852010000             <1>     call    cmp_cmd
2101                                     <1>     ;jnc     make_directory
2102                                     <1>     ; 26/07/2022
2103 00008305 7205                   <1>     jc      short cmp_cmd_rmdir
2104 00008307 E9550A0000             <1>     jmp     make_directory
2105                                     <1>
2106                                     <1> cmp_cmd_rmdir:
2107 0000830C B105                   <1>     mov     cl, 5
2108 0000830E BF[3D2F0100]             <1>     mov     edi, Cmd_Rmdir
2109 00008313 E83F010000             <1>     call    cmp_cmd
2110                                     <1>     ;jnc     delete_directory
2111                                     <1>     ; 26/07/2022
2112 00008318 7205                   <1>     jc      short cmp_cmd_chdir
2113 0000831A E95E0B0000             <1>     jmp     delete_directory
2114                                     <1>
2115                                     <1> cmp_cmd_chdir:
2116 0000831F B105                   <1>     mov     cl, 5
2117 00008321 BF[7A2F0100]             <1>     mov     edi, Cmd_Chdir
2118 00008326 E82C010000             <1>     call    cmp_cmd
2119                                     <1>     ;jc      cmp_cmd_external
2120                                     <1>     ; 26/07/2022
2121 0000832B 7286                   <1>     jc      short cmd_ext
2122                                     <1>
2123 0000832D E92FFFFFFF             <1>     jmp     cd_0
2124                                     <1>
2125                                     <1> c_6:
2126 00008332 80F906                 <1>     cmp     cl, 6
2127                                     <1>     ;ja      c_8
2128                                     <1>     ; 26/07/2022
2129 00008335 72C4                   <1>     jb      short c_5
2130 00008337 7405                   <1>     je      short cmd_6
2131 00008339 E9E4000000             <1>     jmp     c_8
2132                                     <1>
2133                                     <1> cmd_6:
2134                                     <1> cmp_cmd_prompt:
2135 0000833E BF[F92E0100]             <1>     mov     edi, Cmd_Prompt
2136 00008343 E80F010000             <1>     call    cmp_cmd
2137 00008348 722F                   <1>     jc      short cmp_cmd_volume
2138                                     <1> get_prompt_name_fchar:
2139 0000834A AC                     <1>     lodsb
2140 0000834B 3C20                   <1>     cmp     al, 20h
2141 0000834D 74FB                   <1>     je      short get_prompt_name_fchar
2142 0000834F 7713                   <1>     ja      short loc_change_prompt_label
2143                                     <1> default_command_prompt: ; 31/12/2017 ('sysprompt')
2144 00008351 BE[DA2E0100]             <1>     mov     esi, TRDOSPromptLabel
2145 00008356 C7065452444F             <1>     mov     dword [esi], "TRDO"
2146 0000835C 66C746045300             <1>     mov     word [esi+4], "S"
2147                                     <1> loc_cmd_prompt_return:
2148 00008362 C3                     <1>     retn
2149                                     <1>
2150                                     <1> set_command_prompt: ; 31/12/2017 ('sysprompt')
2151 00008363 AC                     <1>     lodsb
2152                                     <1> loc_change_prompt_label:
2153                                     <1>     ;mov     cx, 11
2154                                     <1>     ; 26/07/2022
2155 00008364 29C9                   <1>     sub     ecx, ecx
2156 00008366 B10B                   <1>     mov     cl, 11
2157 00008368 BF[DA2E0100]             <1>     mov     edi, TRDOSPromptLabel
2158                                     <1> put_char_new_prompt_label:
2159 0000836D AA                     <1>     stosb
2160 0000836E AC                     <1>     lodsb
2161 0000836F 3C20                   <1>     cmp     al, 20h
2162 00008371 7202                   <1>     jb      short pass_put_new_prompt_label
2163 00008373 E2F8                   <1>     loop    put_char_new_prompt_label
2164                                     <1> pass_put_new_prompt_label:
2165 00008375 C60700                 <1>     mov     byte [edi], 0
2166 00008378 C3                     <1>     retn
2167                                     <1>
2168                                     <1> cmp_cmd_volume:
2169 00008379 B106                   <1>     mov     cl, 6
2170 0000837B BF[002F0100]             <1>     mov     edi, Cmd_Volume
2171 00008380 E8D2000000             <1>     call    cmp_cmd
2172 00008385 7259                   <1>     jc      short cmp_cmd_attrib
2173                                     <1>
2174                                     <1> cmd_vol1:
2175 00008387 AC                     <1>     lodsb
2176 00008388 3C20                   <1>     cmp     al, 20h
2177 0000838A 7707                   <1>     ja      short cmd_vol2
2178 0000838C A0[62770100]             <1>     mov     al, [Current_Drv]
2179 00008391 EB41                   <1>     jmp     short cmd_vol4
2180                                     <1> cmd_vol2:
2181 00008393 3C41                   <1>     cmp     al, 'A'
2182                                     <1>     ;jb      loc_cmd_failed
2183                                     <1>     ; 26/07/2022
2184 00008395 722D                   <1>     jb      short cmd_vol_failed_1
2185 00008397 3C7A                   <1>     cmp     al, 'z'
2186                                     <1>     ;ja      loc_cmd_failed
2187                                     <1>     ; 26/07/2022
2188 00008399 7731                   <1>     ja      short cmd_vol_failed_2
2189 0000839B 3C5A                   <1>     cmp     al, 'Z'
2190 0000839D 7606                   <1>     jna     short cmd_vol3
2191 0000839F 3C61                   <1>     cmp     al, 'a'
2192                                     <1>     ;jb      loc_cmd_failed
2193                                     <1>     ; 26/07/2022
2194 000083A1 722B                   <1>     jb      short cmd_vol_failed_3
2195 000083A3 24DF                   <1>     and     al, 0DFh
2196                                     <1> cmd_vol3:
2197 000083A5 8A26                   <1>     mov     ah, [esi]
2198 000083A7 80FC3A                 <1>     cmp     ah, '.'
2199 000083AA 0F8594000000             <1>     jne     loc_cmd_failed
2200 000083B0 2C41                   <1>     sub     al, 'A'
2201 000083B2 3A05[7D2E0100]             <1>     cmp     al, [Last_DOS_DiskNo]
2202 000083B8 761A                   <1>     jna     short cmd_vol4
2203                                     <1>
2204 000083BA BE[46310100]             <1>     mov     esi, Msg_Not_Ready_Read_Err
2205 000083BF E9E6E9FFFF             <1>     jmp     print_msg
2206                                     <1>
2207                                     <1> ; 26/07/2022
2208                                     <1> ; (numeric characters and underscore are allowed)
2209                                     <1> cmd_vol_failed_1:
2210                                     <1> ; check for numeric characters
2211 000083C4 3C30                   <1>     cmp     al, '0'
2212 000083C6 7204                   <1>     jb      short cmd_vol_failed_2
2213 000083C8 3C39                   <1>     cmp     al, '9'
2214 000083CA 76D9                   <1>     jna     short cmd_vol3
2215                                     <1> cmd_vol_failed_2:

```

```

2216 000083CC EB76      <1>      jmp      loc_cmd_failed
2217                    <1> cmd_vol_failed_3:
2218 000083CE 3C5F      <1>      cmp     al, '_' ; underline ?
2219 000083D0 74D3      <1>      je      short cmd_vol3 ; is ok..
2220 000083D2 EBF8      <1>      jmp     short cmd_vol_failed_2
2221                    <1>
2222                    <1> cmd_vol4:
2223 000083D4 E8DEFAFFFF <1>      call    print_volume_info
2224                    <1>      ;jc      cd_drive_not_ready
2225                    <1>      ; 26/07/2022
2226 000083D9 7339      <1>      jnc     short cmd_vol5
2227 000083DB E9FAFEFFFF <1>      jmp     cd_drive_not_ready
2228                    <1> ;cmd_vol5:
2229                    <1> ;      retn
2230                    <1>
2231                    <1> cmp_cmd_attrib:
2232 000083E0 B106      <1>      mov     cl, 6
2233 000083E2 BF[2F2F0100] <1>      mov     edi, Cmd_Attrib
2234 000083E7 E86B000000 <1>      call    cmp_cmd
2235                    <1>      ;jnc     set_file_attributes
2236                    <1>      ; 26/07/2022
2237 000083EC 7205      <1>      jc      short cmp_cmd_rename
2238 000083EE E9D80E0000 <1>      jmp     set_file_attributes
2239                    <1>
2240                    <1> cmp_cmd_rename:
2241 000083F3 B106      <1>      mov     cl, 6
2242 000083F5 BF[362F0100] <1>      mov     edi, Cmd_Rename
2243 000083FA E858000000 <1>      call    cmp_cmd
2244                    <1>      ;jnc     rename_file
2245                    <1>      ; 26/07/2022
2246 000083FF 7205      <1>      jc      short cmp_cmd_device
2247 00008401 E9BF100000 <1>      jmp     rename_file
2248                    <1>
2249                    <1> cmp_cmd_device:
2250 00008406 B106      <1>      mov     cl, 6
2251 00008408 BF[6B2F0100] <1>      mov     edi, Cmd_Device
2252 0000840D E845000000 <1>      call    cmp_cmd
2253 00008412 7226      <1>      jc      short cmp_cmd_external
2254                    <1>      ; 26/07/2022
2255                    <1> cmd_vol5:
2256                    <1> cmd_dev:
2257 00008414 C3        <1>      retn
2258                    <1>
2259                    <1> c_7:
2260                    <1> cmp_cmd_devlist:
2261 00008415 BF[722F0100] <1>      mov     edi, Cmd_DevList
2262 0000841A E838000000 <1>      call    cmp_cmd
2263 0000841F 7219      <1>      jc      short cmp_cmd_external
2264                    <1>
2265                    <1> loc_cmd_return:
2266 00008421 C3        <1>      retn
2267                    <1>
2268                    <1> c_8:
2269 00008422 80F908      <1>      cmp     cl, 8
2270 00008425 7713      <1>      ja      short cmp_cmd_external
2271 00008427 72EC      <1>      jb      short c_7
2272                    <1>
2273                    <1> cmp_cmd_longname:
2274 00008429 BF[072F0100] <1>      mov     edi, Cmd_LongName
2275 0000842E E824000000 <1>      call    cmp_cmd
2276                    <1>      ;jnc     get_and_print_longname
2277                    <1>      ; 26/07/2022
2278 00008433 7205      <1>      jc      short cmp_cmd_external
2279 00008435 E95A060000 <1>      jmp     get_and_print_longname
2280                    <1>
2281                    <1> cmp_cmd_external:
2282                    <1>      ; 07/05/2016
2283                    <1>      ; 22/04/2016
2284 0000843A BE[12780100] <1>      mov     esi, CommandBuffer
2285 0000843F E9A6180000 <1>      jmp     loc_run_check_filename
2286                    <1>
2287                    <1> loc_cmd_failed:
2288 00008444 803D[12780100]20 <1>      cmp     byte [CommandBuffer], 20h
2289 00008448 76D4      <1>      jna     short loc_cmd_return
2290 0000844D BE[27310100] <1>      mov     esi, Msg_Bad_Command
2291                    <1>      ; call    print_msg
2292                    <1> ;loc_cmd_return:
2293                    <1>      ;      retn
2294 00008452 E953E9FFFF <1>      jmp     print_msg
2295                    <1>
2296                    <1> cmp_cmd:
2297                    <1>      ; 26/07/2022 (TRDOS 386 kernel v2.0.5)
2298                    <1>      ; 29/01/2016 (TRDOS 386 = TRDOS v2.0)
2299 00008457 BE[12780100] <1>      mov     esi, CommandBuffer
2300                    <1>      ; edi = internal command word (ASCIIIZ)
2301                    <1>      ; ecx = command length (<=8)
2302                    <1> cmp_cmd_1:
2303 0000845C AC        <1>      lodsb
2304 0000845D AE        <1>      scasb
2305 0000845E 750D      <1>      jne     short cmp_cmd_3
2306 00008460 E2FA      <1>      loop    cmp_cmd_1
2307 00008462 AC        <1>      lodsb
2308 00008463 3C20      <1>      cmp     al, 20h
2309 00008465 7703      <1>      ja      short cmp_cmd_2
2310 00008467 30C0      <1>      xor     al, al
2311                    <1>      ; ZF = 1 -> internal command word matches
2312 00008469 C3        <1>      retn
2313                    <1> cmp_cmd_2:
2314                    <1>      ; ZF = 0 (CF = 0) -> external command word
2315 0000846A 58        <1>      pop     eax ; no return to the caller from here
2316 0000846B EBCD      <1>      jmp     short cmp_cmd_external ; 26/07/2022
2317                    <1> cmp_cmd_3:
2318 0000846D F9        <1>      stc
2319                    <1>      ; CF = 1 -> internal command word does not match
2320 0000846E C3        <1>      retn
2321                    <1>
2322                    <1> loc_run_cmd_failed:
2323                    <1>      ; 26/07/2022 (TRDOS 386 kernel v2.0.5)
2324                    <1>      ; 15/03/2016
2325                    <1>      ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
2326                    <1>      ; 07/12/2009 (CMD_INTR.ASM)
2327                    <1>      ; 29/11/2009
2328                    <1>
2329 0000846F E863000000 <1>      call    restore_cdir_after_cmd_fail
2330                    <1>
2331                    <1> loc_run_cmd_failed_cmp_al:
2332                    <1>      ; End of Restore_CDIR code (29/11/2009)
2333                    <1>
2334 00008474 3C01      <1>      cmp     al, 1 ; Bad command or file name
2335 00008476 74CC      <1>      je      short loc_cmd_failed ; 26/07/2022
2336                    <1> loc_run_dir_not_found:
2337 00008478 3C03      <1>      cmp     al, 3
2338 0000847A 750A      <1>      jne     short loc_run_file_notfound_msg
2339                    <1>      ; Path not found (MS-DOS Error Code = 3)

```

```

2340 0000847C BE[89310100] <1> mov esi, Msg_Dir_Not_Found
2341 00008481 E924E9FFFF <1> jmp print_msg
2342 <1>
2343 <1> loc_run_file_notfound_msg:
2344 00008486 3C02 <1> cmp al, 2 ; File not found
2345 00008488 750A <1> jne short loc_run_file_drv_read_err
2346 <1>
2347 <1> loc_print_file_notfound_msg:
2348 0000848A BE[A0310100] <1> mov esi, Msg_File_Not_Found
2349 <1> ;call proc_printmsg
2350 <1> ;retn
2351 0000848F E916E9FFFF <1> jmp print_msg
2352 <1>
2353 <1> loc_run_file_drv_read_err:
2354 <1> ; Err: 17 (Read fault)
2355 00008494 3C11 <1> cmp al, 17 ; Drive not ready or read error
2356 00008496 7404 <1> je short loc_run_file_print_drv_read_err
2357 <1> ;
2358 00008498 3C0F <1> cmp al, 15 ; Drive not ready (or read error)
2359 0000849A 750A <1> jne short loc_run_file_toobig
2360 <1>
2361 <1> loc_run_file_print_drv_read_err:
2362 0000849C BE[46310100] <1> mov esi, Msg_Not_Ready_Read_Err
2363 000084A1 E904E9FFFF <1> jmp print_msg
2364 <1>
2365 <1> loc_run_file_toobig:
2366 000084A6 3C08 <1> cmp al, 8 ; Not enough free memory to load&run file
2367 000084A8 750A <1> jne short loc_run_file_perm_denied
2368 000084AA BE[EB310100] <1> mov esi, Msg_Insufficient_Memory
2369 000084AF E9F6E8FFFF <1> jmp print_msg
2370 <1>
2371 <1> loc_run_file_perm_denied:
2372 <1> ; 29/12/2017
2373 000084B4 3C0B <1> cmp al, ERR_PERM_DENIED ; 11 ; Permission denied
2374 000084B6 750A <1> jne short loc_run_misc_error
2375 000084B8 BE[7F330100] <1> mov esi, Msg_Permission_Denied
2376 000084BD E9E8E8FFFF <1> jmp print_msg
2377 <1>
2378 <1> ; 15/03/2016
2379 <1> print_misc_error_msg:
2380 <1> loc_run_misc_error:
2381 <1> ; AL = Error code
2382 000084C2 E8F4BCFFFF <1> call byteto hex
2383 000084C7 66A3[1F320100] <1> mov [error_code_hex], ax
2384 <1>
2385 000084CD BE[02320100] <1> mov esi, Msg_Error_Code
2386 <1> ;call print_msg
2387 <1> ;retn
2388 000084D2 E9D3E8FFFF <1> jmp print_msg
2389 <1>
2390 <1> restore_cdir_after_cmd_fail:
2391 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
2392 000084D7 50 <1> push eax
2393 000084D8 8A3D[BF7E0100] <1> mov bh, [RUN_CDRV] ; it is set at the beginning
2394 <1> ; of the 'run' command.
2395 000084DE 3A3D[62770100] <1> cmp bh, [Current_Drv]
2396 000084E4 7409 <1> je short loc_run_restore_cdir
2397 000084E6 88FA <1> mov dl, bh
2398 000084E8 E8F3F1FFFF <1> call change_current_drive
2399 000084ED EB19 <1> jmp short loc_run_err_pass_restore_cdir
2400 <1>
2401 <1> loc_run_restore_cdir:
2402 000084EF 803D[7E2E0100]00 <1> cmp byte [Restore_CDIRE], 0
2403 000084F6 7610 <1> jna short loc_run_err_pass_restore_cdir
2404 000084F8 30DB <1> xor bl, bl
2405 000084FA 0FB7F3 <1> movzx esi, bx
2406 000084FD 81C600010900 <1> add esi, Logical_DOSDisks
2407 00008503 E889F2FFFF <1> call restore_current_directory
2408 <1>
2409 <1> loc_run_err_pass_restore_cdir:
2410 00008508 58 <1> pop eax
2411 00008509 C3 <1> retn
2412 <1>
2413 <1> print_directory_list:
2414 <1> ; 02/12/2023 (TRDOS 386 v2.0.7)
2415 <1> ; 07/08/2022
2416 <1> ; 27/07/2022 (TRDOS 386 kernel v2.0.5)
2417 <1> ; 10/02/2016
2418 <1> ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
2419 <1> ; 06/12/2009 ('cmp_cmd_dir')
2420 <1> ;
2421 0000850A 66C705[00800100]00- <1> mov word [AttributesMask], 0800h ; ..except volume names..
2421 00008512 08 <1>
2422 00008513 A0[62770100] <1> mov al, [Current_Drv]
2423 00008518 A2[BF7E0100] <1> mov [RUN_CDRV], al
2424 <1> ; 02/12/2023
2425 0000851D 3C02 <1> cmp al, 2
2426 0000851F 733A <1> jnb short get_dfname_fchar
2427 00008521 A0[21770100] <1> mov al, [TIMER_LOW+1]
2428 00008526 D0E8 <1> shr al, 1 ; 512/18.2 (>= 28 seconds)
2429 00008528 8605[867E0100] <1> xchg al, [P_TIMER] ; 28 seconds
2430 0000852E 3A05[867E0100] <1> cmp al, [P_TIMER]
2431 00008534 7425 <1> je short get_dfname_fchar
2432 00008536 E893F1FFFF <1> call get_media_change_status
2433 0000853B 7205 <1> jc short pdl_chdrv
2434 0000853D 80FC06 <1> cmp ah, 6
2435 00008540 7519 <1> jne short get_dfname_fchar
2436 <1> pdl_chdrv:
2437 00008542 56 <1> push esi
2438 00008543 8A15[BF7E0100] <1> mov dl, [RUN_CDRV]
2439 00008549 E892F1FFFF <1> call change_current_drive
2440 0000854E 5E <1> pop esi
2441 0000854F 730A <1> jnc short get_dfname_fchar
2442 00008551 BE[46310100] <1> mov esi, Msg_Not_Ready_Read_Err
2443 <1> ;call print_msg
2444 <1> ;retn
2445 00008556 E94FE8FFFF <1> jmp print_msg
2446 <1>
2447 <1> get_dfname_fchar:
2448 0000855B AC <1> lodsb
2449 0000855C 3C20 <1> cmp al, 20h
2450 0000855E 74FB <1> je short get_dfname_fchar
2451 <1> ;jb loc_print_dir_call_all
2452 <1> ; 02/12/2023
2453 00008560 7705 <1> ja short get_dfname_fchar_2
2454 00008562 E9E0000000 <1> jmp loc_print_dir_call_all
2455 <1> get_dfname_fchar_2:
2456 00008567 3C2D <1> cmp al, '-'
2457 00008569 753C <1> jne short loc_print_dir_call_flt
2458 <1> get_next_attr_char:
2459 0000856B AC <1> lodsb
2460 0000856C 3C20 <1> cmp al, 20h
2461 0000856E 74FB <1> je short get_next_attr_char
2462 <1> ;jb loc_cmd_failed

```

```

2463      ; 27/07/2022
2464 00008570 7305      jnb     short pdl_1
2465      pdl_0:
2466 00008572 E9CDFEFFFF      jmp     loc_cmd_failed
2467      pdl_1:
2468 00008577 24DF      and     al, 0DFh
2469 00008579 3C44      cmp     al, 'D'; directories only ?
2470 0000857B 750E      jne     short pass_only_directories
2471 0000857D AC      lodsb
2472 0000857E 3C20      cmp     al, 20h
2473      ;ja     loc_cmd_failed
2474      ; 27/07/2022
2475 00008580 77F0      ja     short pdl_0
2476 00008582 800D[00800100]10      or     byte [AttributesMask], 10h ; ..directory..
2477 00008589 EB10      jmp     short get_dfname_fchar_attr
2478      pass_only_directories:
2479 0000858B 3C46      cmp     al, 'F'; files only ?
2480 0000858D 7530      jne     short check_attr_s ; 27/07/2022
2481 0000858F AC      lodsb
2482 00008590 3C20      cmp     al, 20h
2483      ;ja     loc_cmd_failed
2484      ; 27/07/2022
2485 00008592 77DE      ja     short pdl_0
2486 00008594 800D[01800100]10      or     byte [AttributesMask+1], 10h ; ..except directories..
2487      get_dfname_fchar_attr:
2488 0000859B AC      lodsb
2489 0000859C 3C20      cmp     al, 20h
2490 0000859E 74FB      je     short get_dfname_fchar_attr
2491      ;jb     short loc_print_dir_call_all
2492      ; 07/08/2022
2493 000085A0 7305      jnb     short loc_print_dir_call_flg
2494 000085A2 E9A0000000      jmp     loc_print_dir_call_all
2495      loc_print_dir_call_flg:
2496 000085A7 4E      dec     esi
2497 000085A8 BF[02800100]      mov     edi, FindFile_Drv
2498 000085AD E8BC230000      call    parse_path_name
2499 000085B2 7352      jnc     short loc_print_dir_change_drv_1
2500 000085B4 3C01      cmp     al, 1
2501      ;ja     loc_run_cmd_failed
2502      ; 27/07/2022
2503 000085B6 764E      jna     short loc_print_dir_change_drv_1
2504      pdl_2:
2505 000085B8 E9B2FEFFFF      jmp     loc_run_cmd_failed
2506
2507      ; 27/07/2022
2508      check_attr_s_cap:
2509 000085BD 24DF      and     al, 0DFh
2510      check_attr_s:
2511 000085BF 3C53      cmp     al, 's'
2512 000085C1 7510      jne     short pass_attr_s
2513 000085C3 800D[00800100]04      or     byte [AttributesMask], 4 ; system
2514 000085CA AC      lodsb
2515 000085CB 3C20      cmp     al, 20h
2516 000085CD 74CC      je     short get_dfname_fchar_attr
2517 000085CF 7276      jb     short loc_print_dir_call_all
2518 000085D1 24DF      and     al, 0DFh
2519      pass_attr_s:
2520 000085D3 3C48      cmp     al, 'H'
2521 000085D5 7510      jne     short pass_attr_h
2522 000085D7 800D[00800100]02      or     byte [AttributesMask], 2 ; hidden
2523      pass_attr_shr:
2524 000085DE AC      lodsb
2525 000085DF 3C20      cmp     al, 20h
2526 000085E1 74B8      je     short get_dfname_fchar_attr
2527 000085E3 7262      jb     short loc_print_dir_call_all
2528 000085E5 EBD6      jmp     short check_attr_s_cap
2529      pass_attr_h:
2530 000085E7 3C52      cmp     al, 'R'
2531 000085E9 7509      jne     short pass_attr_r
2532 000085EB 800D[00800100]01      or     byte [AttributesMask], 1 ; read only
2533 000085F2 EBEA      jmp     short pass_attr_shr
2534      pass_attr_r:
2535 000085F4 3C41      cmp     al, 'A'
2536      ;jne     loc_cmd_failed
2537      ; 27/07/2022
2538 000085F6 7405      je     short pass_attr_a
2539 000085F8 E947FEFFFF      jmp     loc_cmd_failed
2540      pass_attr_a:
2541 000085FD 800D[00800100]20      or     byte [AttributesMask], 20h ; archive
2542 00008604 EBD8      jmp     short pass_attr_shr
2543
2544      ; 07/08/2022
2545      loc_print_dir_change_drv_1:
2546 00008606 8A15[02800100]      mov     dl, [FindFile_Drv]
2547      loc_print_dir_change_drv_2:
2548 0000860C 3A15[BF7E0100]      cmp     dl, [RUN_CDRV]
2549 00008612 7407      je     short loc_print_dir_change_directory
2550 00008614 E8C7F0FFFF      call    change_current_drive
2551      ;jc     loc_run_cmd_failed
2552      ; 27/07/2022
2553 00008619 729D      jc     short pdl_2
2554      loc_print_dir_change_directory:
2555 0000861B 803D[03800100]20      cmp     byte [FindFile_Directory], 20h ; 0 or 20h ?
2556 00008622 7619      jna     short pass_print_dir_change_directory
2557
2558 00008624 FE05[7E2E0100]      inc     byte [Restore_CDIR]
2559 0000862A BE[03800100]      mov     esi, FindFile_Directory
2560 0000862F 30E4      xor     ah, ah ; CD_COMMAND sign -> 0
2561 00008631 E8A51D0000      call    change_current_directory
2562      ;jc     loc_run_cmd_failed
2563      ; 27/07/2022
2564 00008636 7280      jc     short pdl_2
2565
2566      loc_print_dir_change_prompt_dir_string:
2567 00008638 E8C71C0000      call    change_prompt_dir_string
2568
2569      pass_print_dir_change_directory:
2570 0000863D BE[44800100]      mov     esi, FindFile_Name
2571 00008642 803E20      cmp     byte [esi], 20h ; ; 0 or 20h ?
2572 00008645 7706      ja     short loc_print_dir_call
2573
2574      loc_print_dir_call_all:
2575 00008647 C7062A2E2A00      mov     dword [esi], '.*'
2576      loc_print_dir_call:
2577 0000864D E82D000000      call    print_directory
2578
2579 00008652 8A15[BF7E0100]      mov     dl, [RUN_CDRV] ; it is set at the beginning
2580 00008658 3A15[62770100]      cmp     dl, [Current_Drv]
2581 0000865E 7405      je     short loc_print_dir_call_restore_cdir_retn
2582      ;call    change_current_drive
2583      ;retn
2584      ; 27/07/2022
2585 00008660 E97BF0FFFF      jmp     change_current_drive
2586

```

```

2587 <1> loc_print_dir_call_restore_cdir_retn:
2588 00008665 803D[7E2E0100]00 <1> cmp byte [Restore_CDIR], 0
2589 0000866C 7610 <1> jna short pass_print_dir_call_restore_cdir_retn
2590 <1>
2591 0000866E BE00010900 <1> mov esi, Logical_DOSDisks
2592 00008673 31C0 <1> xor eax, eax
2593 00008675 88D4 <1> mov ah, dl
2594 00008677 01C6 <1> add esi, eax
2595 <1>
2596 <1> ;call restore_current_directory
2597 <1> ; 27/07/2022
2598 00008679 E913F1FFFF <1> jmp restore_current_directory
2599 <1>
2600 <1> pass_print_dir_call_restore_cdir_retn:
2601 0000867E C3 <1> retn
2602 <1>
2603 <1> print_directory:
2604 <1> ; 27/07/2022 (TRDOS 386 Kernel v2.0.5)
2605 <1> ; 13/05/2016
2606 <1> ; 11/02/2016
2607 <1> ; 10/02/2016
2608 <1> ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
2609 <1> ; 30/10/2010 ('proc_print_directory')
2610 <1> ; 19/09/2009
2611 <1> ; 2005
2612 <1> ; INPUT ->
2613 <1> ; ESI = Asciz File/Dir Name Address
2614 <1>
2615 0000867F 56 <1> push esi
2616 <1>
2617 00008680 29C0 <1> sub eax, eax
2618 <1>
2619 00008682 66A3[8C800100] <1> mov [Dir_Count], ax ; 0
2620 00008688 66A3[8A800100] <1> mov [File_Count], ax ; 0
2621 0000868E A3[8E800100] <1> mov [Total_FSize], eax ; 0
2622 <1>
2623 00008693 E828E7FFFF <1> call clear_screen
2624 <1>
2625 00008698 31C9 <1> xor ecx, ecx
2626 0000869A 8A2D[62770100] <1> mov ch, [Current_Drv] ; DirBuff_Drv - 'A'
2627 000086A0 A0[63770100] <1> mov al, [Current_Dir_Drv]
2628 000086A5 A2[44300100] <1> mov [Dir_Drive_Name], al
2629 000086AA BE00010900 <1> mov esi, Logical_DOSDisks
2630 000086AF 01CE <1> add esi, ecx
2631 <1>
2632 000086B1 E856F9FFFF <1> call move_volume_name_and_serial_no
2633 000086B6 730C <1> jnc short print_dir_strlen_check
2634 <1>
2635 000086B8 5E <1> pop esi
2636 000086B9 8A3D[CE760100] <1> mov bh, [ptty] ; [ACTIVE_PAGE]
2637 <1> ;call beeper
2638 <1> ;retn
2639 000086BF E9019DFFFF <1> jmp beeper ; beep ! and return
2640 <1>
2641 <1> print_dir_strlen_check:
2642 000086C4 BE[65770100] <1> mov esi, Current_Dir_Root
2643 000086C9 BF[E1300100] <1> mov edi, Dir_Str_Root
2644 <1>
2645 <1> ;xor ecx, ecx
2646 000086CE 8A0D[C1770100] <1> mov cl, [Current_Dir_StrLen]
2647 000086D4 FEC1 <1> inc cl
2648 000086D6 80F940 <1> cmp cl, 64
2649 000086D9 760D <1> jna short pass_print_dir_strlen_shorting
2650 000086DB 46 <1> inc esi
2651 000086DC 01CE <1> add esi, ecx
2652 000086DE 83EE40 <1> sub esi, 64
2653 000086E1 47 <1> inc edi
2654 000086E2 B82E2E2E20 <1> mov eax, '... '
2655 000086E7 AB <1> stosd
2656 <1>
2657 <1> pass_print_dir_strlen_shorting:
2658 000086E8 F3A4 <1> rep movsb
2659 <1>
2660 000086EA BE[37300100] <1> mov esi, Dir_Drive_Str
2661 000086EF E8B6E6FFFF <1> call print_msg
2662 <1>
2663 000086F4 BE[96300100] <1> mov esi, Vol_Serial_Header
2664 000086F9 E8ACE6FFFF <1> call print_msg
2665 <1>
2666 000086FE BE[D6300100] <1> mov esi, Dir_Str_Header
2667 00008703 E8A2E6FFFF <1> call print_msg
2668 <1>
2669 00008708 BE[05390100] <1> mov esi, next2line
2670 0000870D E898E6FFFF <1> call print_msg
2671 <1>
2672 <1> loc_print_dir_first_file:
2673 00008712 C605[A1800100]10 <1> mov byte [PrintDir_RowCounter], 16
2674 00008719 66A1[00800100] <1> mov ax, [AttributesMask]
2675 0000871F 5E <1> pop esi
2676 <1>
2677 00008720 E845020000 <1> call find_first_file
2678 <1> ;jc loc_dir_ok
2679 <1> ; 27/07/2022
2680 00008725 7305 <1> jnc short loc_dfname_use_this
2681 00008727 E97B010000 <1> jmp loc_dir_ok
2682 <1>
2683 <1> loc_dfname_use_this:
2684 <1> ; bl = File Attributes (bh = Long Name Entry Length)
2685 0000872C F6C310 <1> test bl, 10h ; Is it a directory?
2686 0000872F 741B <1> jz short loc_not_dir
2687 <1>
2688 00008731 66FF05[8C800100] <1> inc word [Dir_Count]
2689 00008738 89F2 <1> mov edx, esi ; FindFile_DirEntry address
2690 0000873A BE[25320100] <1> mov esi, Type_Dir ; '<DIR>'
2691 0000873F BF[3C320100] <1> mov edi, Dir_Or_FileSize
2692 <1> ; move 10 bytes
2693 00008744 A5 <1> movsd
2694 00008745 A5 <1> movsd
2695 00008746 66A5 <1> movsw
2696 00008748 89D6 <1> mov esi, edx
2697 0000874A EB36 <1> jmp short loc_dir_attribute
2698 <1>
2699 <1> loc_not_dir:
2700 0000874C 66FF05[8A800100] <1> inc word [File_Count]
2701 00008753 0105[8E800100] <1> add [Total_FSize], eax
2702 <1>
2703 00008759 B90A000000 <1> mov ecx, 10 ; 32 bit divisor
2704 0000875E 89CF <1> mov edi, ecx
2705 00008760 81C7[3C320100] <1> add edi, Dir_Or_FileSize
2706 <1> loc_dir_rdivide:
2707 00008766 29D2 <1> sub edx, edx
2708 00008768 F7F1 <1> div ecx ; remainder in dl (< 10)
2709 0000876A 80C230 <1> add dl, '0' ; to make visible (ascii)
2710 0000876D 4F <1> dec edi

```



```

2711 0000876E 8817      <1>    mov     [edi], dl
2712 00008770 21C0      <1>    and     eax, eax
2713 00008772 75F2      <1>    jnz     short loc_dir_rdivide
2714          <1>
2715          <1> loc_dir_fill_space:
2716 00008774 81FF[3C320100] <1>    cmp     edi, Dir_Or_FileSize
2717 0000877A 7606      <1>    jna     short loc_dir_attribute
2718 0000877C 4F         <1>    dec     edi
2719 0000877D C60720    <1>    mov     byte [edi], 20h
2720 00008780 EBF2      <1>    jmp     short loc_dir_fill_space
2721          <1>
2722          <1> loc_dir_attribute:
2723 00008782 C705[47320100]2020- <1>    mov     dword [File_Attribute], 20202020h
2723 0000878A 2020      <1>
2724          <1>
2725 0000878C 80FB20    <1>    cmp     bl, 20h ; Is it an archive file?
2726 0000878F 7207      <1>    jb     short loc_dir_pass_arch
2727 00008791 C605[4A320100]41    <1>    mov     byte [File_Attribute+3], 'A'
2728          <1>
2729          <1> loc_dir_pass_arch:
2730 00008798 80E307    <1>    and     bl, 7
2731 0000879B 7428      <1>    jz     short loc_dir_file_name
2732 0000879D 88DF      <1>    mov     bh, bl
2733 0000879F 80E303    <1>    and     bl, 3
2734 000087A2 38DF      <1>    cmp     bh, bl
2735 000087A4 7607      <1>    jna     short loc_dir_pass_s
2736 000087A6 C605[47320100]53    <1>    mov     byte [File_Attribute], 's'
2737          <1>
2738          <1> loc_dir_pass_s:
2739 000087AD 80E302    <1>    and     bl, 2
2740 000087B0 7407      <1>    jz     short loc_dir_pass_h
2741 000087B2 C605[48320100]48    <1>    mov     byte [File_Attribute+1], 'H'
2742          <1> loc_dir_pass_h:
2743 000087B9 80E701    <1>    and     bh, 1
2744 000087BC 7407      <1>    jz     short loc_dir_file_name
2745 000087BE C605[49320100]52    <1>    mov     byte [File_Attribute+2], 'R'
2746          <1> loc_dir_file_name:
2747          <1> ;mov     bx, [esi+18h] ; Date
2748          <1> ;mov     dx, [esi+16h] ; Time
2749 000087C5 8B5E16    <1>    mov     ebx, [esi+16h]
2750 000087C8 89F1      <1>    mov     ecx, esi ; FindFile_DirEntry address
2751 000087CA BF[2F320100] <1>    mov     edi, File_Name
2752          <1> ; move 8 bytes
2753 000087CF A5         <1>    movsd
2754 000087D0 A5         <1>    movsd
2755 000087D1 C60720    <1>    mov     byte [edi], 20h
2756 000087D4 47         <1>    inc     edi
2757          <1> ; move 3 bytes
2758 000087D5 66A5      <1>    movsw
2759 000087D7 A4         <1>    movsb
2760 000087D8 89CE      <1>    mov     esi, ecx
2761          <1>
2762          <1> Dir_Time_start:
2763          <1> ;mov     ax, dx ; Time
2764 000087DA 6689D8    <1>    mov     ax, bx
2765 000087DD 66C1E805  <1>    shr     ax, 5 ; shift right 5 times
2766 000087E1 6683E03F  <1>    and     ax, 000011111b ; Minute Mask
2767 000087E5 D40A      <1>    aam ; Q([AL]/10)->AH
2768          <1> ; R([AL]/10)->AL
2769          <1> ; [AL]+[AH]= Minute as BCD
2770 000087E7 660D3030 <1>    or      ax, '00' ; Convert to ASCII
2771 000087EB 86C4      <1>    xchg    ah, al
2772 000087ED 66A3[5A320100] <1>    mov     [File_Minute], ax
2773          <1>
2774          <1> ;mov     al, dh
2775 000087F3 88F8      <1>    mov     al, bh
2776 000087F5 C0E803    <1>    shr     al, 3 ; shift right 3 times
2777 000087F8 D40A      <1>    aam ; [AL]+[AH]= Hours as BCD
2778 000087FA 660D3030 <1>    or      ax, '00'
2779 000087FE 86C4      <1>    xchg    ah, al
2780 00008800 66A3[57320100] <1>    mov     [File_Hour], ax
2781          <1>
2782 00008806 C1EB10    <1>    shr     ebx, 16 ; BX = Date
2783          <1>
2784          <1> Dir_Date_start:
2785          <1> ;mov     ax, bx ; Date
2786          <1> ; 27/07/2022
2787 00008809 89D8      <1>    mov     eax, ebx
2788 0000880B 6683E01F  <1>    and     ax, 00011111b ; Day Mask
2789 0000880F D40A      <1>    aam ; Q([AL]/10)->AH
2790          <1> ; R([AL]/10)->AL
2791          <1> ; [AL]+[AH]= Day as BCD
2792 00008811 660D3030 <1>    or      ax, '00' ; Convert to ASCII
2793 00008815 86E0      <1>    xchg    al, ah
2794          <1>
2795 00008817 66A3[4C320100] <1>    mov     [File_Day], ax
2796          <1>
2797          <1> ;mov     ax, bx
2798          <1> ; 27/07/2022
2799 0000881D 89D8      <1>    mov     eax, ebx
2800          <1> ;shr     ax, 5 ; shift right 5 times
2801 0000881F C1E805    <1>    shr     eax, 5
2802 00008822 6683E00F  <1>    and     ax, 00001111b ; Month Mask
2803 00008826 D40A      <1>    aam
2804 00008828 660D3030 <1>    or      ax, '00'
2805 0000882C 86C4      <1>    xchg    ah, al
2806 0000882E 66A3[4F320100] <1>    mov     [File_Month], ax
2807          <1>
2808          <1> ;mov     ax, bx
2809          <1> ; 27/07/2022
2810 00008834 89D8      <1>    mov     eax, ebx
2811          <1> ;shr     ax, 9
2812 00008836 C1E809    <1>    shr     eax, 9
2813 00008839 6683E07F  <1>    and     ax, 01111111b ; Result = Year - 1980
2814 0000883D 6605BC07  <1>    add     ax, 1980
2815          <1>
2816 00008841 B10A      <1>    mov     cl, 10
2817 00008843 F6F1      <1>    div     cl ; Q -> AL, R -> AH
2818 00008845 80CC30    <1>    or      ah, '0'
2819 00008848 8825[55320100] <1>    mov     [File_Year+3], ah
2820 0000884E D40A      <1>    aam
2821 00008850 86C4      <1>    xchg    ah, al
2822 00008852 80CC30    <1>    or      ah, '0' ; Convert to ASCII
2823 00008855 8825[54320100] <1>    mov     [File_Year+2], ah
2824 0000885B D40A      <1>    aam
2825 0000885D 86E0      <1>    xchg    al, ah
2826 0000885F 660D3030 <1>    or      ax, '00'
2827 00008863 66A3[52320100] <1>    mov     [File_Year], ax
2828          <1>
2829          <1> loc_show_line:
2830 00008869 56         <1>    push    esi
2831 0000886A BE[2F320100] <1>    mov     esi, File_Name
2832 0000886F E836E5FFFF <1>    call    print_msg
2833 00008874 BE[07390100] <1>    mov     esi, nextline

```

```

2834 00008879 E82CE5FFFF <1> call print_msg
2835 0000887E 5E <1> pop esi
2836 <1>
2837 0000887F FE0D[A1800100] <1> dec byte [PrintDir_RowCounter]
2838 00008885 740C <1> jz short pause_dir_scroll ; 27/07/2022
2839 <1>
2840 <1> loc_next_entry:
2841 00008887 E88B010000 <1> call find_next_file
2842 <1> ;jnc loc_dfname_use_this
2843 <1> ; 27/07/2022
2844 0000888C 7219 <1> jc short loc_dir_ok
2845 0000888E E999FEFFFF <1> jmp loc_dfname_use_this
2846 <1>
2847 <1> ; 27/07/2022
2848 <1> pause_dir_scroll:
2849 00008893 28E4 <1> sub ah, ah
2850 00008895 E88286FFFF <1> call int16h
2851 0000889A 3C1B <1> cmp al, 1Bh
2852 0000889C 7409 <1> je short loc_dir_ok
2853 0000889E C605[A1800100]10 <1> mov byte [PrintDir_RowCounter], 16 ; Reset counter
2854 000088A5 EBE0 <1> jmp short loc_next_entry
2855 <1>
2856 <1> loc_dir_ok:
2857 000088A7 B90A000000 <1> mov ecx, 10
2858 <1> ;mov ax, [Dir_Count]
2859 <1> ; 27/07/2022
2860 000088AC 0FB705[8C800100] <1> movzx eax, word [Dir_Count]
2861 000088B3 BF70320100 <1> mov edi, Decimal_Dir_Count
2862 <1> ;cmp ax, cx ; 10
2863 <1> ; 27/07/2022
2864 000088B8 39C8 <1> cmp eax, ecx
2865 000088BA 7216 <1> jb short pass_ddc
2866 000088BC 47 <1> inc edi
2867 000088BD 6683F864 <1> cmp ax, 100
2868 000088C1 720F <1> jb short pass_ddc
2869 000088C3 47 <1> inc edi
2870 000088C4 663DE803 <1> cmp ax, 1000
2871 000088C8 7208 <1> jb short pass_ddc
2872 000088CA 47 <1> inc edi
2873 000088CB 663D1027 <1> cmp ax, 10000
2874 000088CF 7201 <1> jb short pass_ddc
2875 000088D1 47 <1> inc edi
2876 <1> pass_ddc:
2877 000088D2 886F01 <1> mov [edi+1], ch ; 0
2878 <1> loc_ddc_rediv:
2879 000088D5 31D2 <1> xor edx, edx
2880 <1> ;div cx ; 10
2881 <1> ; 27/07/2022
2882 000088D7 F7F1 <1> div ecx
2883 000088D9 80C230 <1> add dl, '0'
2884 000088DC 8817 <1> mov [edi], dl
2885 000088DE 4F <1> dec edi
2886 <1> ;or ax, ax
2887 <1> ; 27/07/2022
2888 000088DF 09C0 <1> or eax, eax
2889 000088E1 75F2 <1> jnz short loc_ddc_rediv
2890 <1>
2891 000088E3 66A1[8A800100] <1> mov ax, [File_Count]
2892 000088E9 BF5F320100 <1> mov edi, Decimal_File_Count
2893 <1> ;cmp ax, cx ; 10
2894 <1> ; 27/07/2022
2895 000088EE 39C8 <1> cmp eax, ecx ; 10
2896 000088F0 7216 <1> jb short pass_dfc
2897 000088F2 47 <1> inc edi
2898 000088F3 6683F864 <1> cmp ax, 100
2899 000088F7 720F <1> jb short pass_dfc
2900 000088F9 47 <1> inc edi
2901 000088FA 663DE803 <1> cmp ax, 1000
2902 000088FE 7208 <1> jb short pass_dfc
2903 00008900 47 <1> inc edi
2904 00008901 663D1027 <1> cmp ax, 10000
2905 00008905 7201 <1> jb short pass_dfc
2906 00008907 47 <1> inc edi
2907 <1> pass_dfc:
2908 <1> ;mov cx, 10
2909 00008908 886F01 <1> mov [edi+1], ch ; 00
2910 <1> loc_dfc_rediv:
2911 <1> ;xor dx, dx
2912 <1> ;xor dl, dl
2913 <1> ;div cx
2914 <1> ; 27/07/022
2915 0000890B 31D2 <1> xor edx, edx
2916 0000890D F7F1 <1> div ecx
2917 0000890F 80C230 <1> add dl, '0'
2918 00008912 8817 <1> mov [edi], dl
2919 00008914 4F <1> dec edi
2920 <1> ;or ax, ax
2921 <1> ; 27/07/2022
2922 00008915 09C0 <1> or eax, eax
2923 00008917 75F2 <1> jnz short loc_dfc_rediv
2924 <1>
2925 00008919 BF[A0800100] <1> mov edi, TFS_Dec_End
2926 <1> ;mov byte [edi], 0
2927 0000891E A1[8E800100] <1> mov eax, [Total_FSize]
2928 <1> ;mov ecx, 10
2929 <1> rediv_tfs_hex:
2930 00008923 29D2 <1> sub edx, edx ; 27/07/2022
2931 <1> ;sub dl, dl
2932 00008925 F7F1 <1> div ecx
2933 00008927 80C230 <1> add dl, '0'
2934 0000892A 4F <1> dec edi
2935 0000892B 8817 <1> mov [edi], dl
2936 0000892D 21C0 <1> and eax, eax
2937 0000892F 75F2 <1> jnz short rediv_tfs_hex
2938 <1>
2939 00008931 893D[92800100] <1> mov [TFS_Dec_Begin], edi
2940 00008937 BE5D320100 <1> mov esi, Decimal_File_Count_Header
2941 0000893C E869E4FFFF <1> call print_msg
2942 00008941 BE65320100 <1> mov esi, str_files
2943 00008946 E85FE4FFFF <1> call print_msg
2944 0000894B BE76320100 <1> mov esi, str_dirs
2945 00008950 E855E4FFFF <1> call print_msg
2946 00008955 8B35[92800100] <1> mov esi, [TFS_Dec_Begin]
2947 0000895B E84AE4FFFF <1> call print_msg
2948 00008960 BE87320100 <1> mov esi, str_bytes
2949 <1> ;call print_msg
2950 <1> ;ret
2951 <1> ; 27/07/2022
2952 00008965 E940E4FFFF <1> jmp print_msg
2953 <1>
2954 <1> find_first_file:
2955 <1> ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
2956 <1> ; 11/02/2016
2957 <1> ; 10/02/2016

```

```

2958      <1>      ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
2959      <1>      ; 09/10/2011
2960      <1>      ; 17/09/2009
2961      <1>      ; 2005
2962      <1>      ; INPUT ->
2963      <1>      ;     ESI = ASCIIZ File/Dir Name Address (in Current Directory)
2964      <1>      ;     AL = Attributes AND mask (The AND result must be equal to AL)
2965      <1>      ;         bit 0 = Read Only
2966      <1>      ;         bit 1 = Hidden
2967      <1>      ;         bit 2 = System
2968      <1>      ;         bit 3 = Volume Label
2969      <1>      ;         bit 4 = Directory
2970      <1>      ;         bit 5 = Archive
2971      <1>      ;         bit 6 = Reserved, must be 0
2972      <1>      ;         bit 7 = Reserved, must be 0
2973      <1>      ;     AH = Attributes Negative AND mask (The AND result must be ZERO)
2974      <1>      ;
2975      <1>      ; OUTPUT ->
2976      <1>      ;     CF = 1 -> Error, Error Code in EAX (AL)
2977      <1>      ;     CF = 0 ->
2978      <1>      ;     ESI = Directory Entry (FindFile_DirEntry) Location
2979      <1>      ;     EDI = Directory Buffer Directory Entry Location
2980      <1>      ;     EAX = File Size
2981      <1>      ;     BL = Attributes of The File/Directory
2982      <1>      ;     BH = Long Name Yes/No Status (>0 is YES)
2983      <1>      ;     DX > 0 : Ambiguous filename chars are used
2984      <1>      ;
2985      <1>      ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
2986      <1>
2987      0000896A 66A3[52800100] <1>      mov     [FindFile_AttributesMask], ax
2988      00008970 BF[54800100] <1>      mov     edi, FindFile_DirEntry ; TR-DOS Fullfilename formatted buffer
2989      00008975 31C0 <1>      xor     eax, eax
2990      <1>      ;mov     ecx, 11
2991      <1>      ; 28/07/2022
2992      00008977 31C9 <1>      xor     ecx, ecx
2993      00008979 B10B <1>      mov     cl, 11
2994      0000897B F3AB <1>      rep     stosd    ; 44 bytes
2995      <1>      ;stosw     ; +2 bytes
2996      <1>
2997      0000897D BF[44800100] <1>      mov     edi, FindFile_Name ; FFF structure, offset 66
2998      00008982 39FE <1>      cmp     esi, edi
2999      00008984 7408 <1>      je      short loc_fff_mfn_ok
3000      00008986 89FA <1>      mov     edx, edi
3001      <1>      ; move 13 bytes
3002      00008988 A5 <1>      movsd
3003      00008989 A5 <1>      movsd
3004      0000898A A5 <1>      movsd
3005      0000898B AA <1>      stosb
3006      0000898C 89D6 <1>      mov     esi, edx
3007      <1>      loc_fff_mfn_ok:
3008      0000898E BF[F37F0100] <1>      mov     edi, Dir_Entry_Name ; Dir Entry Format File Name
3009      00008993 E80C1F0000 <1>      call    convert_file_name
3010      00008998 89FE <1>      mov     esi, edi ; offset Dir_Entry_Name
3011      <1>
3012      0000899A 66A1[52800100] <1>      mov     ax, [FindFile_AttributesMask]
3013      <1>      ;xor     ecx, ecx
3014      000089A0 30C9 <1>      xor     cl, cl
3015      000089A2 E8511C0000 <1>      call    locate_current_dir_file
3016      000089A7 726D <1>      jc      short loc_fff_retn
3017      <1>      ; EDI = Directory Entry
3018      <1>      ; EBX = Directory Buffer Entry Index/Number
3019      <1>
3020      <1>      loc_fff_fnf_ln_check:
3021      000089A9 30ED <1>      xor     ch, ch
3022      000089AB 80F60F <1>      xor     dh, 0Fh
3023      000089AE 7408 <1>      jz      short loc_fff_longname_yes
3024      000089B0 882D[51800100] <1>      mov     [FindFile_LongNameYes], ch ; 0
3025      000089B6 EB0C <1>      jmp     short loc_fff_longname_no
3026      <1>
3027      <1>      loc_fff_longname_yes:
3028      <1>      ;inc     byte [FindFile_LongNameYes]
3029      000089B8 8A0D[5E7F0100] <1>      mov     cl, [LFN_EntryLength]
3030      000089BE 880D[51800100] <1>      mov     [FindFile_LongNameEntryLength], cl ; FindFile_LongNameYes
3031      <1>
3032      <1>      loc_fff_longname_no:
3033      <1>      ;mov     bx, [DirBuff_CurrentEntry]
3034      000089C4 66891D[7C800100] <1>      mov     [FindFile_DirEntryNumber], bx
3035      <1>      ;mov     dx, ax ; Ambiguous Filename chars used sign > 0
3036      <1>      ; 28/07/2022
3037      000089CB 89C2 <1>      mov     edx, eax
3038      <1>
3039      000089CD A0[62770100] <1>      mov     al, [Current_Drv]
3040      000089D2 A2[02800100] <1>      mov     [FindFile_Drv], al
3041      <1>
3042      000089D7 A1[5C770100] <1>      mov     eax, [Current_Dir_FCluster]
3043      000089DC A3[74800100] <1>      mov     [FindFile_DirFirstCluster], eax
3044      <1>
3045      000089E1 A1[8E7E0100] <1>      mov     eax, [DirBuff_Cluster]
3046      000089E6 A3[78800100] <1>      mov     [FindFile_DirCluster], eax
3047      <1>
3048      000089EB 66FF05[7E800100] <1>      inc     word [FindFile_MatchCounter]
3049      <1>
3050      000089F2 89FB <1>      mov     ebx, edi
3051      000089F4 89FE <1>      mov     esi, edi
3052      000089F6 BF[54800100] <1>      mov     edi, FindFile_DirEntry
3053      000089FB 89F8 <1>      mov     eax, edi
3054      000089FD B108 <1>      mov     cl, 8
3055      000089FF F3A5 <1>      rep     movsd
3056      00008A01 89C6 <1>      mov     esi, eax
3057      00008A03 89DF <1>      mov     edi, ebx
3058      <1>
3059      00008A05 A1[70800100] <1>      mov     eax, [FindFile_DirEntry+28] ; File Size
3060      <1>
3061      00008A0A 8A1D[5F800100] <1>      mov     bl, [FindFile_DirEntry+11] ; File Attributes
3062      00008A10 8A3D[51800100] <1>      mov     bh, [FindFile_LongNameYes]
3063      <1>
3064      <1>      ;mov     cx, [DirBuff_EntryCounter]
3065      <1>      ;mov     [FindFile_DirEntryNumber], cx
3066      <1>      ;mov     cx, [FindFile_DirEntryNumber]
3067      <1>      ; ecx = 0
3068      <1>
3069      <1>      loc_fff_retn:
3070      00008A16 C3 <1>      retn
3071      <1>
3072      <1>      find_next_file:
3073      <1>      ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
3074      <1>      ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
3075      <1>      ; 15/10/2016
3076      <1>      ; 10/02/2016
3077      <1>      ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
3078      <1>      ; 06/02/2011
3079      <1>      ; 17/09/2009
3080      <1>      ; 2005
3081      <1>      ; INPUT ->

```

```

3082      ;      NONE, Find First File Parameters
3083      ;      OUTPUT ->
3084      ;      CF = 1 -> Error, Error Code in EAX (AL)
3085      ;      CF = 0 ->
3086      ;      ESI = Directory Entry (FindFile_DirEntry) Location
3087      ;      EDI = Directory Buffer Directory Entry Location
3088      ;      EAX = File Size
3089      ;      BL = Attributes of The File/Directory
3090      ;      BH = Long Name Yes/No Status (>0 is YES)
3091      ;      DX > 0 : Ambiguous filename chars are used
3092      ;
3093      ;      (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
3094      ;
3095 00008A17 66833D[7E800100]00      cmp     word [FindFile_MatchCounter], 0
3096 00008A1F 7707                  ja      short loc_start_search_next_file
3097
3098      loc_fnf_stc_retn:
3099 00008A21 F9                  stc
3100      loc_fnf_ax12h_retn:
3101 00008A22 B80C000000          mov     eax, 12 ; No More files
3102      ;loc_fnf_retn:
3103 00008A27 C3                  retn
3104
3105      loc_start_search_next_file:
3106 00008A28 668B1D[7C800100]      mov     bx, [FindFile_DirEntryNumber]
3107      ;inc     bx
3108      ; 28/07/2022
3109 00008A2F 43                  inc     ebx
3110 00008A30 663B1D[8C7E0100]      cmp     bx, [DirBuff_LastEntry]
3111 00008A37 7719                  ja      short loc_cont_search_next_file
3112
3113      loc_fnf_search:
3114 00008A39 BE[F37F0100]          mov     esi, Dir_Entry_Name
3115 00008A3E 66A1[52800100]        mov     ax, [FindFile_AttributesMask]
3116      ;xor     cx, cx
3117      ; 28/07/2022
3118 00008A44 31C9              xor     ecx, ecx
3119 00008A46 E8881C0000          call    find_directory_entry
3120      ;jnc     loc_fff_fnf_ln_check
3121      ; 28/07/2022
3122 00008A4B 7205              jc      short loc_cont_search_next_file
3123 00008A4D E957FFFFFF          jmp     loc_fff_fnf_ln_check
3124
3125      loc_cont_search_next_file:
3126 00008A52 31DB              xor     ebx, ebx
3127 00008A54 8A3D[62770100]        mov     bh, [Current_Drv]
3128 00008A5A BE00010900          mov     esi, Logical_DOSDisks
3129 00008A5F 01DE              add     esi, ebx
3130
3131 00008A61 803D[60770100]00      cmp     byte [Current_Dir_Level], 0
3132      ; 19/12/2025
3133      %if 0
3134      jna      short loc_fnf_check_FAT_type
3135
3136      cmp     byte [esi+LD_FATType], 1
3137      jnb      short loc_fnf_ax12h_retn
3138
3139      jmp     short loc_fnf_check_next_cluster
3140      %else
3141      ; 19/12/2025
3142 00008A68 7706              ja      short loc_fnf_check_next_cluster
3143      %endif
3144
3145      loc_fnf_check_FAT_type:
3146 00008A6A 807E0303          cmp     byte [esi+LD_FATType], 3
3147 00008A6E 72B2              jnb     short loc_fnf_ax12h_retn
3148
3149      loc_fnf_check_next_cluster:
3150 00008A70 A1[8E7E0100]          mov     eax, [DirBuff_Cluster]
3151 00008A75 E8FF340000          call    get_next_cluster
3152 00008A7A 7306              jnc     short loc_fnf_load_next_dir_cluster
3153 00008A7C 09C0              or      eax, eax
3154 00008A7E 74A1              jz      short loc_fnf_stc_retn
3155      ;mov     eax, 17 ;Drive not ready or read error
3156 00008A80 F5                  cmc
3157      ;stc
3158 00008A81 C3                  retn
3159
3160      loc_fnf_load_next_dir_cluster:
3161 00008A82 E8AE360000          call    load_FAT_sub_directory
3162 00008A87 72F8              jc      short loc_fnf_retn
3163      ;xor     bx, bx
3164      ; 28/07/2022
3165 00008A89 31DB              xor     ebx, ebx
3166 00008A8B 66891D[7C800100]        mov     [FindFile_DirEntryNumber], bx
3167 00008A92 EBA5              jmp     short loc_fnf_search
3168
3169      get_and_print_longname:
3170      ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
3171      ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
3172      ; 16/10/2016
3173      ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)
3174      ; 24/01/2010
3175      ; 17/10/2009 (CMD_INTR.ASM, 'cmp_cmd_longname')
3176      get_longname_fchar:
3177 00008A94 803E20          cmp     byte [esi], 20h
3178 00008A97 7701              ja      short loc_find_longname
3179      ;jb      short loc_longname_retn
3180      ;inc     esi
3181      ;je      short get_longname_fchar
3182      ;loc_longname_retn:
3183      retn
3184      loc_find_longname:
3185 00008A9A E86C1F0000          call    find_longname
3186 00008A9F 731D              jnc     short loc_print_longname
3187
3188      or      al, al
3189 00008AA3 7412              jz      short loc_longname_not_found
3190
3191      ; 16/10/2016 (15h -> 15, 17)
3192 00008AA5 3C0F              cmp     al, 15
3193      ;je      cd_drive_not_ready ; drive not ready
3194      ; 28/07/2022
3195 00008AA7 7505              jne     short loc_fln_err2
3196      loc_fln_err1:
3197 00008AA9 E92CF8FFFF          jmp     cd_drive_not_ready
3198      loc_fln_err2:
3199      ; or
3200 00008AAE 3C11              cmp     al, 17 ; read error
3201      ;je      cd_drive_not_ready
3202      ; 28/07/2022
3203      ;je      short loc_fln_err1
3204
3205      loc_ln_file_dir_not_found:

```

```

3206 00008AB0 BE[B2310100] <1> mov esi, Msg_File_Directory_Not_Found
3207 <1> ;;call print_msg
3208 <1> ;;retn
3209 <1> ;jmp print_msg
3210 <1> ; 28/07/2022
3211 00008AB5 EB1F <1> jmp short loc_lfn_err3
3212 <1>
3213 <1> loc_longname_not_found:
3214 00008AB7 BE[D1310100] <1> mov esi, Msg_LongName_Not_Found
3215 <1> ;;call print_msg
3216 <1> ;;retn
3217 <1> ;jmp print_msg
3218 <1> ; 28/07/2022
3219 00008ABC EB18 <1> jmp short loc_lfn_err3
3220 <1>
3221 <1> loc_print_longname:
3222 <1> ;mov esi, LongFileName
3223 00008ABE BF[62780100] <1> mov edi, TextBuffer
3224 00008AC3 57 <1> push edi
3225 <1>
3226 <1> ; 19/12/2025
3227 <1> %if 0
3228 <1> cmp al, 0
3229 <1> ja short loc_print_longname_1
3230 <1> loc_print_FS_longname: ; Singlix FS (64 byte ASCIIZ file name)
3231 <1> lodsb
3232 <1> stosb
3233 <1> or al, al
3234 <1> jnz short loc_print_FS_longname
3235 <1> jmp short loc_print_longname_2
3236 <1> %endif
3237 <1> ;
3238 <1> loc_print_longname_1: ; MS windows long name (UNICODE chars)
3239 00008AC4 66AD <1> lodsw
3240 00008AC6 AA <1> stosb
3241 00008AC7 08C0 <1> or al, al
3242 00008AC9 75F9 <1> jnz short loc_print_longname_1
3243 <1> ;
3244 <1> loc_print_longname_2:
3245 00008ACB 5E <1> pop esi
3246 00008ACC E8D9E2FFFF <1> call print_msg
3247 00008AD1 BE[07390100] <1> mov esi, nextline
3248 <1> loc_lfn_err3:
3249 <1> ;call print_msg
3250 <1> ;retn
3251 00008AD6 E9CFE2FFFF <1> jmp print_msg
3252 <1>
3253 <1> show_file:
3254 <1> ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
3255 <1> ; 07/08/2022
3256 <1> ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
3257 <1> ; 18/02/2016
3258 <1> ; 17/02/2016
3259 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
3260 <1> ; 13/09/2011 (CMD_INTR.ASM, 'cmp_cmd_show')
3261 <1> ; 08/11/2009
3262 <1>
3263 <1> loc_show_parse_path_name:
3264 00008ADB BF[02800100] <1> mov edi, FindFile_Drv
3265 00008AE0 E8891E0000 <1> call parse_path_name
3266 <1> ;jc loc_cmd_failed
3267 <1> ; 28/07/2022
3268 00008AE5 7305 <1> jnc short loc_show_check_filename_exists
3269 <1> show_file_err1:
3270 00008AE7 E958F9FFFF <1> jmp loc_cmd_failed
3271 <1>
3272 <1> loc_show_check_filename_exists:
3273 00008AEC BE[44800100] <1> mov esi, FindFile_Name
3274 00008AF1 803E20 <1> cmp byte [esi], 20h
3275 <1> ;jna loc_cmd_failed
3276 <1> ; 28/07/2022
3277 00008AF4 76F1 <1> jna short show_file_err1
3278 <1>
3279 <1> ; 15/02/2016 (invalid file name check)
3280 00008AF6 E8CE010000 <1> call check_filename
3281 00008AFB 730A <1> jnc short loc_show_change_drv
3282 <1>
3283 00008AFD BE[9D320100] <1> mov esi, Msg_invalid_name_chars
3284 00008B02 E9A3E2FFFF <1> jmp print_msg
3285 <1>
3286 <1> loc_show_change_drv:
3287 00008B07 8A35[62770100] <1> mov dh, [Current_Drv]
3288 00008B0D 8835[BF7E0100] <1> mov [RUN_CDRV], dh
3289 00008B13 8A15[02800100] <1> mov dl, [FindFile_Drv]
3290 00008B19 38F2 <1> cmp dl, dh
3291 00008B1B 740C <1> je short loc_show_change_directory
3292 00008B1D E8BEEBFFFF <1> call change_current_drive
3293 <1> ;jc loc_file_rw_cmd_failed
3294 <1> ;jc loc_run_cmd_failed
3295 <1> ; 28/07/2022
3296 00008B22 7305 <1> jnc short loc_show_change_directory
3297 <1> show_file_err2:
3298 00008B24 E946F9FFFF <1> jmp loc_run_cmd_failed
3299 <1>
3300 <1> loc_show_change_directory:
3301 00008B29 803D[03800100]20 <1> cmp byte [FindFile_Directory], 20h
3302 00008B30 7614 <1> jna short loc_findload_showfile
3303 <1>
3304 00008B32 FE05[7E2E0100] <1> inc byte [Restore_CDIR]
3305 00008B38 BE[03800100] <1> mov esi, FindFile_Directory
3306 00008B3D 30E4 <1> xor ah, ah ; CD_COMMAND sign -> 0
3307 00008B3F E897180000 <1> call change_current_directory
3308 <1> ;jc loc_file_rw_cmd_failed
3309 <1> ;jc loc_run_cmd_failed
3310 <1> ; 28/07/2022
3311 00008B44 72DE <1> jc short show_file_err2
3312 <1>
3313 <1> ;loc_show_change_prompt_dir_string:
3314 <1> ;call change_prompt_dir_string
3315 <1>
3316 <1> loc_findload_showfile:
3317 <1> ; 15/02/2016
3318 00008B46 BE[44800100] <1> mov esi, FindFile_Name
3319 00008B48 BF[F37F0100] <1> mov edi, Dir_Entry_Name ; Dir Entry Format File Name
3320 00008B50 E84F1D0000 <1> call convert_file_name
3321 00008B55 89FE <1> mov esi, edi ; offset Dir_Entry_Name
3322 <1>
3323 00008B57 28C0 <1> sub al, al ; Attrib AND mask = 0
3324 <1> ; Directory attribute : 10h
3325 <1> ; Volume name attribute: 8h
3326 00008B59 B418 <1> mov ah, 00011000b ; 18h (Attrib NAND, AND --> zero mask)
3327 <1> ;
3328 <1> ;xor cx, cx
3329 <1> ; 28/07/2022

```

```

3330 00008B5B 31C9      <1>      xor     ecx, ecx
3331 00008B5D E8961A0000 <1>      call    locate_current_dir_file
3332                      <1>      ;;jc     loc_file_rw_cmd_failed
3333                      <1>      ;jc     loc_run_cmd_failed
3334                      <1>      ; 28/07/2022
3335 00008B62 72C0      <1>      jc      short show_file_err2
3336                      <1>
3337                      <1> loc_show_load_file:
3338                      <1>      ; EDI = Directory Entry
3339 00008B64 668B4714    <1>      mov     ax, [edi+DirEntry_FstClusHI] ; First Cluster High word
3340 00008B68 C1E010     <1>      shl     eax, 16
3341 00008B6B 668B471A    <1>      mov     ax, [edi+DirEntry_FstClusLO] ; First Cluster Low word
3342 00008B6F A3AC800100] <1>      mov     [Show_Cluster], eax
3343 00008B74 8B471C     <1>      mov     eax, [edi+DirEntry_FileSize] ; File Size
3344 00008B77 21C0      <1>      and     eax, eax ; Empty file !
3345                      <1>      ;jz     end_of_show_file
3346                      <1>      ; 28/07/2022
3347 00008B79 7505      <1>      jnz     short loc_show_load_file_set_size
3348 00008B7B E994000000 <1>      jmp     end_of_show_file
3349                      <1>
3350                      <1> loc_show_load_file_set_size: ; 28/07/2022
3351 00008B80 A3B0800100] <1>      mov     [Show_FileSize], eax
3352 00008B85 31C0      <1>      xor     eax, eax
3353 00008B87 A3B4800100] <1>      mov     [Show_FilePointer], eax ; 0
3354 00008B8C 66A3B8800100] <1>      mov     [Show_ClusterPointer], ax ; 0
3355 00008B92 29DB      <1>      sub     ebx, ebx
3356 00008B94 8A3D62770100] <1>      mov     bh, [Current_Drv]
3357 00008B9A BE00010900 <1>      mov     esi, Logical_DOSDisks
3358 00008B9F 01DE      <1>      add     esi, ebx
3359 00008BA1 8935A8800100] <1>      mov     [Show_LDDDT], esi ; Logical DOS Drv Description Table addr
3360                      <1>
3361                      <1> ; 19/12/2025
3362                      <1> %if 0
3363                      <1>      cmp     byte [esi+LD_FATType], 0
3364                      <1>      ja      short loc_show_calculate_cluster_size
3365                      <1>      ; Singlix FS
3366                      <1>      ; First Cluster Number is FDT number (in compatibility buffer)
3367                      <1>      mov     edx, [Show_Cluster] ; Compatibility dir. buffer value (FDT)
3368                      <1>      mov     [Show_FDT], edx
3369                      <1>      ; 19/12/2025
3370                      <1>      ;xor     eax, eax
3371                      <1>      mov     [Show_Cluster], eax ; Sector index = 0
3372                      <1>      ; (next time it will be 1)
3373                      <1> %endif
3374                      <1>
3375                      <1> loc_show_calculate_cluster_size:
3376                      <1>      ;mov     bx, [esi+LD_BPB+BPB_BytsPerSec] ; FAT 12-16-32 (512)
3377                      <1>      ; 19/12/2025
3378 00008BA7 66BB0002    <1>      mov     bx, 512
3379                      <1>      ; BX = 512 = [esi+LD_FS_BytesPerSec] ; Singlix FS
3380 00008BAB 8A4613     <1>      mov     al, [esi+LD_BPB+BPB_SecPerClust] ; FAT 12-16-32 (<= 128)
3381                      <1>      ; AL = 1 = [esi+LD_FS_Reserved2] ; SectPerClust for Singlix FS
3382 00008BAE F7E3      <1>      mul     ebx
3383                      <1>
3384                      <1>      ;cmp     eax, 65536 ; non-compatible (very big) cluster size
3385                      <1>      ;ja      short end_of_show_file
3386 00008BB0 66A3BA800100] <1>      mov     [Show_ClusterSize], ax
3387                      <1>
3388                      <1> loc_start_show_file:
3389 00008BB6 BE07390100] <1>      mov     esi, nextline
3390 00008BBB E8EAE1FFFF <1>      call    print_msg
3391                      <1>
3392 00008BC0 A1AC800100] <1>      mov     eax, [Show_Cluster]
3393 00008BC5 C605BC800100]17 <1>      mov     byte [Show_RowCount], 23
3394                      <1>
3395                      <1>      ; 17/02/2016
3396 00008BCC 8B35A8800100] <1>      mov     esi, [Show_LDDDT]
3397                      <1>
3398                      <1>      ; 07/08/2022
3399                      <1> loc_show_next_cluster:
3400                      <1>      ; 15/02/2016
3401 00008BD2 BB00000700 <1>      mov     ebx, cluster_Buffer ; 70000h (for current TRDOS 386 version)
3402                      <1>      ; ESI = Logical DOS drv description table address
3403 00008BD7 E88F350000 <1>      call    read_cluster
3404                      <1>      ;;jc     loc_file_rw_cmd_failed
3405                      <1>      ;jc     loc_run_cmd_failed
3406                      <1>      ; 07/08/2022
3407 00008BDC 7305      <1>      jnc     short loc_show_nc_rc_ok
3408 00008BDE E98CF8FFFF <1>      jmp     loc_run_cmd_failed
3409                      <1> loc_show_nc_rc_ok:
3410                      <1>      xor     ebx, ebx
3411                      <1> loc_show_next_byte:
3412 00008BE5 803DBC800100]00 <1>      cmp     byte [Show_RowCount], 0
3413 00008BEC 7512      <1>      jne     short pass_show_wait_for_key
3414 00008BEE 30E4      <1>      xor     ah, ah
3415 00008BF0 E82783FFFF <1>      call    int16h
3416 00008BF5 3C1B      <1>      cmp     al, 1Bh
3417                      <1>      ;jne     short pass_exit_show
3418                      <1>      ; 28/07/2022
3419 00008BF7 741B      <1>      je      short end_of_show_file
3420                      <1>
3421                      <1> ;end_of_show_file:
3422                      <1> ;pass_show_file:
3423                      <1>      ; mov     esi, nextline
3424                      <1>      ; call    print_msg
3425                      <1>      ; jmp     loc_file_rw_restore_retn
3426                      <1>
3427                      <1> pass_exit_show:
3428 00008BF9 C605BC800100]14 <1>      mov     byte [Show_RowCount], 20
3429                      <1> pass_show_wait_for_key:
3430                      <1>      add     ebx, cluster_Buffer
3431 00008C06 8A03      <1>      mov     al, [ebx]
3432 00008C08 3C0D      <1>      cmp     al, 0Dh
3433                      <1>      ;jne     loc_show_check_tab_space
3434                      <1>      ; 28/07/2022
3435 00008C0A 7417      <1>      je      short loc_show_dec_row_count
3436 00008C0C EB7E      <1>      jmp     loc_show_check_tab_space
3437                      <1>
3438                      <1>      ; 07/08/2022
3439                      <1> loc_show_next:
3440                      <1>      ;and     bx, bx ; 65536 -> 0
3441                      <1>      ; 28/07/2022
3442 00008C0E 21DB      <1>      and     ebx, ebx
3443 00008C10 75D3      <1>      jnz     short loc_show_next_byte
3444 00008C12 EBBE      <1>      jmp     short loc_show_next_cluster
3445                      <1>
3446                      <1>      ; 19/12/2025
3447                      <1>      ; 28/07/2022
3448                      <1> end_of_show_file:
3449                      <1> pass_show_file:
3450 00008C14 BE07390100] <1>      mov     esi, nextline
3451 00008C19 E88CE1FFFF <1>      call    print_msg
3452 00008C1E E915010000 <1>      jmp     loc_file_rw_restore_retn
3453                      <1>

```

```

3454      <1> loc_show_dec_row_count:      ; 28/07/2022
3455 00008C23 FE0D[BC800100]      <1> dec byte [Show_RowCount]
3456      <1> pass_show_dec_rowcount:
3457 00008C29 B307      <1> mov bl, 7 ; (light gray character color, black background)
3458 00008C2B 8A3D[CE760100]      <1> mov bh, [ACTIVE_PAGE] ; [ptty]
3459 00008C31 E8A596FFFF      <1> call _write_tty
3460      <1> loc_show_check_eof:
3461 00008C36 FF05[B4800100]      <1> inc dword [Show_FilePointer]
3462 00008C3C A1[B4800100]      <1> mov eax, [Show_FilePointer]
3463 00008C41 3B05[B0800100]      <1> cmp eax, [Show_FileSize]
3464 00008C47 73CB      <1> jnb short end_of_show_file
3465 00008C49 66FF05[B8800100]      <1> inc word [Show_ClusterPointer]
3466 00008C50 0FB71D[B8800100]      <1> movzx ebx, word [Show_ClusterPointer]
3467      <1>
3468      <1> ; 17/02/2016
3469      <1> ; (sector boundary -9 bits- check, 512 = 0)
3470 00008C57 66F7C3FF01      <1> test bx, 1FFh ; 1 to 511
3471 00008C5C 7587      <1> jnz short loc_show_next_byte
3472      <1>
3473      <1> ; 16/02/2016
3474 00008C5E 8B35[A8800100]      <1> mov esi, [Show_LDDDT]
3475      <1>
3476      <1> ; 19/12/2025
3477      <1> %if 0
3478      <1> cmp byte [esi+LD_FATType], 0
3479      <1> ja short loc_show_check_fat_cluster_size
3480      <1>
3481      <1> ; Singlix FS
3482      <1> ; 1 sector, more... (cluster size = 1 sector)
3483      <1> mov eax, [Show_Cluster]
3484      <1> inc eax
3485      <1> mov [Show_Cluster], eax
3486      <1>
3487      <1> ; 07/08/2022
3488      <1> jmp short loc_show_next
3489      <1>
3490      <1> ; 28/07/2022
3491      <1> end_of_show_file:
3492      <1> pass_show_file:
3493      <1> mov esi, nextline
3494      <1> call print_msg
3495      <1> jmp loc_file_rw_restore_retn
3496      <1> %endif
3497      <1>
3498      <1> loc_show_check_fat_cluster_size:
3499      <1> ; 17/02/2016
3500 00008C64 66B1D[BA800100]      <1> cmp bx, [Show_ClusterSize] ; cluster size in bytes
3501      <1> ;jb short loc_show_next_byte ; 28/07/2022
3502      <1> ; 07/08/2022
3503 00008C6B 7305      <1> jnb short loc_show_file_cluster_ok
3504 00008C6D E973FFFFFF      <1> jmp loc_show_next_byte
3505      <1>
3506      <1> loc_show_file_cluster_ok:
3507 00008C72 66C705[B8800100]00-      <1> mov word [Show_ClusterPointer], 0
3507 00008C7A 00      <1>
3508      <1>
3509 00008C7B A1[AC800100]      <1> mov eax, [Show_Cluster]
3510      <1> ;mov esi, [Show_LDDDT]
3511      <1> loc_show_get_next_cluster:
3512 00008C80 E8F4320000      <1> call get_next_cluster
3513      <1> ;jc loc_file_rw_cmd_failed
3514      <1> ;jc loc_run_cmd_failed
3515      <1> ; 28/07/2022
3516 00008C85 7338      <1> jnc short loc_show_update_ccluster
3517 00008C87 E9E3F7FFFF      <1> jmp loc_run_cmd_failed
3518      <1>
3519      <1> loc_show_check_tab_space:
3520 00008C8C 3C09      <1> cmp al, 09h
3521      <1> ;jne short pass_show_dec_rowcount ; 28/07/2022
3522      <1> ; 07/08/2022
3523 00008C8E 7402      <1> je short loc_show_put_tab_space
3524 00008C90 EB97      <1> jmp pass_show_dec_rowcount
3525      <1> loc_show_put_tab_space:
3526 00008C92 8A3D[CE760100]      <1> mov bh, [ACTIVE_PAGE] ; [ptty]
3527 00008C98 E8DC92FFFF      <1> call get_cpos
3528      <1> ; dl = cursor column
3529 00008C9D 80E207      <1> and dl, 7 ; 18/02/2016
3530      <1> ;shr bh, 1 ; [ACTIVE_PAGE]
3531 00008CA0 8A3D[CE760100]      <1> mov bh, [ACTIVE_PAGE]
3532 00008CA6 B307      <1> mov bl, 7 ; color attribute
3533      <1> loc_show_put_space_chars:
3534 00008CA8 B020      <1> mov al, 20h ; space
3535      <1> ;mov bh, [ACTIVE_PAGE] ; [ptty]
3536      <1> ;mov bl, 7 ; color attribute
3537      <1> ;push dx
3538 00008CAA 52      <1> push edx ; 29/12/2017
3539 00008CAB E82B96FFFF      <1> call _write_tty
3540 00008CB0 5A      <1> pop edx ; 29/12/2017
3541      <1> ;pop dx
3542      <1> ; 18/02/2016
3543 00008CB1 80FA07      <1> cmp dl, 7
3544      <1> ;jnb short loc_show_check_eof ; 28/07/2022
3545      <1> ; 07/08/2022
3546 00008CB4 7205      <1> jb short loc_show_next_tab_space
3547 00008CB6 E97BFFFFFF      <1> jmp loc_show_check_eof
3548      <1> loc_show_next_tab_space:
3549      <1> inc dl
3550 00008CBD EBE9      <1> jmp short loc_show_put_space_chars
3551      <1>
3552      <1> loc_show_update_ccluster:
3553 00008CBF A3[AC800100]      <1> mov [Show_Cluster], eax
3554 00008CC4 E909FFFFFF      <1> jmp loc_show_next_cluster
3555      <1>
3556      <1> check_filename:
3557      <1> ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
3558      <1> ; 10/10/2016
3559      <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
3560      <1> ; 07/08/2010 (FILE.ASM, 'proc_check_filename')
3561      <1> ; 10/07/2010
3562      <1> ; Derived from 'proc_check_filename'
3563      <1> ; in the old TRDOS.ASM (09/02/2005).
3564      <1> ;
3565      <1> ; INPUT ->
3566      <1> ; ESI = Dot File Name Location
3567      <1> ; OUTPUT ->
3568      <1> ; cf = 1 -> error code in AL
3569      <1> ; AL = ERR_INV_FILE_NAME (=26)
3570      <1> ; Invalid file name chars
3571      <1> ; cf = 0 -> valid file name
3572      <1> ;
3573      <1> ; (EAX, ECX, EDI will be changed)
3574      <1>
3575      <1> check_invalid_filename_chars:
3576      <1> ; 28/07/2022 (TRDOS 386 kernel v2.0.5)

```

```

3577      <1>      ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
3578      <1>      ; 10/07/2010 (FILE.ASM, 'proc_check_invalid_filename_chars')
3579      <1>      ; 10/02/2010
3580      <1>      ; Derived from 'proc_check_invalid_filename_chars'
3581      <1>      ; in the old TRDOS.ASM (09/02/2005).
3582      <1>      ;
3583      <1>      ; INPUT ->
3584      <1>      ;     ESI = ASCIIZ FileName
3585      <1>      ; OUTPUT ->
3586      <1>      ;     cf = 1 -> invalid
3587      <1>      ;     cf = 0 -> valid
3588      <1>      ;
3589      <1>      ;(EAX, ECX, EDI will be changed)
3590      <1>
3591      00008CC9 56      <1>      push     esi
3592      <1>
3593      00008CCA BF[862F0100] <1>      mov      edi, invalid_fname_chars
3594      00008CCF AC      <1>      lodsb
3595      <1>      check_filename_next_char:
3596      00008CD0 B914000000 <1>      mov      ecx, sizeInvFnChars
3597      00008CD5 BF[862F0100] <1>      mov      edi, invalid_fname_chars
3598      <1>      loc_scan_invalid_filename_char:
3599      00008CDA AE      <1>      scasb
3600      00008CDB 741E      <1>      je       short loc_invalid_filename_stc
3601      00008CDD E2FB      <1>      loop    loc_scan_invalid_filename_char
3602      00008CDF AC      <1>      lodsb
3603      00008CE0 3C1F      <1>      cmp      al, 1Fh ; 20h and above
3604      00008CE2 77EC      <1>      ja       short check_filename_next_char
3605      <1>
3606      <1>      check_filename_dot:
3607      00008CE4 8B3424      <1>      mov      esi, [esp]
3608      <1>
3609      00008CE7 B421      <1>      mov      ah, 21h
3610      <1>      ;mov     ecx, 8
3611      <1>      ; 28/07/2022
3612      00008CE9 29C9      <1>      sub      ecx, ecx
3613      00008CEB B108      <1>      mov      cl, 8
3614      <1>      loc_check_filename_next_char:
3615      00008CED AC      <1>      lodsb
3616      00008CEE 3C2E      <1>      cmp      al, 2Eh
3617      00008CF0 7511      <1>      jne      short pass_check_fn_dot_check
3618      <1>      loc_check_filename_ext_0:
3619      00008CF2 AC      <1>      lodsb
3620      00008CF3 38E0      <1>      cmp      al, ah ; 21h
3621      00008CF5 7205      <1>      jb       short loc_invalid_filename
3622      00008CF7 3C2E      <1>      cmp      al, 2Eh
3623      00008CF9 7519      <1>      jne      short loc_check_filename_ext_1
3624      <1>
3625      <1>      loc_invalid_filename_stc:
3626      <1>      loc_check_fn_stc_rtn:
3627      00008CFB F9      <1>      stc
3628      <1>      loc_invalid_filename:
3629      <1>      ; 10/10/2016 (0Bh -> 26)
3630      00008CFC B81A000000 <1>      mov      eax, ERR_INV_FILE_NAME ; (=26)
3631      <1>      ; Invalid file name chars
3632      <1>      loc_check_fn_rtn:
3633      00008D01 5E      <1>      pop      esi
3634      00008D02 C3      <1>      retn
3635      <1>
3636      <1>      pass_check_fn_dot_check:
3637      00008D03 38E0      <1>      cmp      al, ah ; 21h
3638      00008D05 7224      <1>      jb       short loc_check_fn_clc_rtn
3639      00008D07 E2E4      <1>      loop    loc_check_filename_next_char
3640      00008D09 AC      <1>      lodsb
3641      00008D0A 38E0      <1>      cmp      al, ah ; 21h
3642      00008D0C 721D      <1>      jb       short loc_check_fn_clc_rtn
3643      00008D0E 3C2E      <1>      cmp      al, 2Eh
3644      00008D10 75E9      <1>      jne      short loc_check_fn_stc_rtn
3645      00008D12 EBDE      <1>      jmp      short loc_check_filename_ext_0
3646      <1>
3647      <1>      loc_check_filename_ext_1:
3648      00008D14 AC      <1>      lodsb
3649      00008D15 38E0      <1>      cmp      al, ah ; 21h
3650      00008D17 7212      <1>      jb       short loc_check_fn_clc_rtn
3651      00008D19 3C2E      <1>      cmp      al, 2Eh
3652      00008D1B 74DE      <1>      je       short loc_check_fn_stc_rtn
3653      00008D1D AC      <1>      lodsb
3654      00008D1E 38E0      <1>      cmp      al, ah ; 21h
3655      00008D20 7209      <1>      jb       short loc_check_fn_clc_rtn
3656      00008D22 3C2E      <1>      cmp      al, 2Eh
3657      00008D24 74D5      <1>      je       short loc_check_fn_stc_rtn
3658      00008D26 AC      <1>      lodsb
3659      00008D27 38E0      <1>      cmp      al, ah ; 21h
3660      00008D29 73D0      <1>      jnb      short loc_check_fn_stc_rtn
3661      <1>
3662      <1>      loc_check_fn_clc_rtn:
3663      00008D2B 5E      <1>      pop      esi
3664      00008D2C F8      <1>      clc
3665      00008D2D C3      <1>      retn
3666      <1>
3667      <1>      loc_print_deleted_message:
3668      00008D2E BE[72330100] <1>      mov      esi, Msg_Deleted
3669      00008D33 E872E0FFFF <1>      call     print_msg
3670      <1>
3671      <1>      ;clc
3672      <1>
3673      <1>      loc_file_rw_restore_retn:
3674      <1>      ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
3675      <1>      ; 28/02/2010 (CMD_INTR.ASM)
3676      <1>      loc_file_rw_cmd_failed:
3677      00008D38 9C      <1>      pushf
3678      00008D39 E899F7FFFF <1>      call     restore_cdir_after_cmd_fail
3679      00008D3E 9D      <1>      popf
3680      00008D3F 720D      <1>      jc       short loc_file_rw_check_write_fault
3681      00008D41 C3      <1>      retn
3682      <1>
3683      <1>      loc_permission_denied:
3684      <1>      ; 27/02/2016
3685      00008D42 BE[7F330100] <1>      mov      esi, Msg_Permission_Denied
3686      00008D47 E85EE0FFFF <1>      call     print_msg
3687      00008D4C EBEA      <1>      jmp      short loc_file_rw_restore_retn
3688      <1>
3689      <1>      loc_file_rw_check_write_fault:
3690      <1>      ;cmp     al, 1Dh ; Write Fault
3691      00008D4E 3C12      <1>      cmp      al, 18 ; 05/11/2016
3692      <1>      ;jne     loc_run_cmd_failed_cmp_al
3693      <1>      ; 28/07/2022
3694      00008D50 7405      <1>      je       short loc_file_rw_fault
3695      00008D52 E91DF7FFFF <1>      jmp      loc_run_cmd_failed_cmp_al
3696      <1>
3697      <1>      loc_file_rw_fault:
3698      00008D57 BE[67310100] <1>      mov      esi, Msg_Not_Ready_Write_Err
3699      <1>      ;call    print_msg
3700      <1>      ;retn

```



```

3701 00008D5C E949E0FFFF <1> jmp print_msg
3702 <1>
3703 <1> make_directory:
3704 <1> ; 28/07/2022 (TRDOS 386 Kernel v2.0.5)
3705 <1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
3706 <1> ; 12/03/2011 (CMD_INTR.ASM, 'cmp_cmd_mkdir')
3707 <1> ; 14/08/2010
3708 <1> ; 10/07/2010
3709 <1> ; 29/11/2009
3710 <1> ;
3711 <1> get_mkdir_fchar:
3712 <1> ; esi = directory name
3713 00008D61 803E20 <1> cmp byte [esi], 20h
3714 00008D64 7701 <1> ja short loc_mkdir_parse_path_name
3715 <1>
3716 <1> loc_mkdir_nodirname_retn:
3717 00008D66 C3 <1> retn
3718 <1>
3719 <1> loc_mkdir_parse_path_name:
3720 00008D67 BF[02800100] <1> mov edi, FindFile_Drv
3721 00008D6C E8FD1B0000 <1> call parse_path_name
3722 <1> ;jc loc_cmd_failed
3723 <1> ; 28/07/2022
3724 00008D71 7305 <1> jnc short loc_mkdir_check_dirname_exists
3725 <1> loc_mkdir_cmd_failed:
3726 00008D73 E9CCF6FFFF <1> jmp loc_cmd_failed
3727 <1>
3728 <1> loc_mkdir_check_dirname_exists:
3729 00008D78 BE[44800100] <1> mov esi, FindFile_Name
3730 00008D7D 803E20 <1> cmp byte [esi], 20h
3731 <1> ;jna loc_cmd_failed
3732 <1> ; 28/07/2022
3733 00008D80 76F1 <1> jna short loc_mkdir_cmd_failed
3734 00008D82 8935[C0800100] <1> mov [DelFile_FNPointer], esi
3735 00008D88 E83CFFFFFF <1> call check_filename
3736 00008D8D 725B <1> jc short loc_mkdir_invalid_dir_name_chars
3737 <1>
3738 <1> loc_mkdir_drv:
3739 00008D8F 8A35[62770100] <1> mov dh, [Current_Drv]
3740 00008D95 8835[B7E0100] <1> mov [RUN_CDRV], dh
3741 <1>
3742 00008D9B 8A15[02800100] <1> mov dl, [FindFile_Drv]
3743 00008DA1 38F2 <1> cmp dl, dh
3744 00008DA3 7409 <1> je short loc_mkdir_change_directory
3745 <1>
3746 00008DA5 E836E9FFFF <1> call change_current_drive
3747 <1> ;jc loc_file_rw_cmd_failed
3748 <1> ; 28/07/2022
3749 00008DAA 7302 <1> jnc short loc_mkdir_change_directory
3750 00008DAC EB8A <1> jmp loc_file_rw_cmd_failed
3751 <1>
3752 <1> loc_mkdir_change_directory:
3753 00008DAE 803D[03800100]20 <1> cmp byte [FindFile_Directory], 20h
3754 00008DB5 7614 <1> jna short loc_mkdir_find_directory
3755 <1>
3756 00008DB7 FE05[7E2E0100] <1> inc byte [Restore_CDIR]
3757 00008DBD BE[03800100] <1> mov esi, FindFile_Directory
3758 00008DC2 30E4 <1> xor ah, ah ; CD_COMMAND sign -> 0
3759 00008DC4 E812160000 <1> call change_current_directory
3760 00008DC9 7226 <1> jc short loc_mkdir_check_error_code
3761 <1>
3762 <1> ;loc_mkdir_change_prompt_dir_string:
3763 <1> ;call change_prompt_dir_string
3764 <1>
3765 <1> loc_mkdir_find_directory:
3766 <1> ;mov esi, FindFile_Name
3767 00008DCB 8B35[C0800100] <1> mov esi, [DelFile_FNPointer]
3768 <1> ;xor eax, eax
3769 00008DD1 6631C0 <1> xor ax, ax ; any name (dir, file, volume)
3770 00008DD4 E891FBFFFF <1> call find_first_file
3771 00008DD9 7216 <1> jc short loc_mkdir_check_error_code
3772 <1>
3773 <1> loc_mkdir_directory_found:
3774 00008DDB BE[CA320100] <1> mov esi, Msg_Name_Exists
3775 <1> loc_mkdir_inv_dname_chrs_msg:
3776 00008DE0 E8C5DFFFFFFF <1> call print_msg
3777 00008DE5 E94EFFFFFFF <1> jmp loc_file_rw_restore_retn
3778 <1>
3779 <1> loc_mkdir_invalid_dir_name_chars:
3780 00008DEA BE[9D320100] <1> mov esi, Msg_invalid_name_chars
3781 <1> ;call print_msg
3782 <1> ;jmp loc_file_rw_restore_retn
3783 <1> ; 28/07/2022
3784 00008DEF EBEF <1> jmp short loc_mkdir_inv_dname_chrs_msg
3785 <1>
3786 <1> loc_mkdir_check_error_code:
3787 00008DF1 3C02 <1> cmp al, 2
3788 <1> ;je short loc_mkdir_directory_not_found
3789 00008DF3 7406 <1> je short loc_mkdir_ask_for_yes_no
3790 00008DF5 F9 <1> stc
3791 00008DF6 E93DFFFFFFF <1> jmp loc_file_rw_cmd_failed
3792 <1>
3793 <1> loc_mkdir_directory_not_found:
3794 <1> loc_mkdir_ask_for_yes_no:
3795 00008DFB BE[EB320100] <1> mov esi, Msg_DoYouWantMkdir
3796 00008E00 E8A5DFFFFFFF <1> call print_msg
3797 00008E05 8B35[C0800100] <1> mov esi, [DelFile_FNPointer]
3798 00008E0B E89ADFFFFFFF <1> call print_msg
3799 00008E10 BE[0A330100] <1> mov esi, Msg_YesNo
3800 00008E15 E890DFFFFFFF <1> call print_msg
3801 <1>
3802 00008E1A C605[14330100]20 <1> mov byte [Y_N_nextline], 20h
3803 <1>
3804 <1> loc_mkdir_ask_again:
3805 00008E21 30E4 <1> xor ah, ah
3806 00008E23 E8F480FFFF <1> call int16h
3807 00008E28 3C1B <1> cmp al, 1Bh
3808 <1> ;je short loc_do_not_make_directory
3809 00008E2A 743B <1> je short loc_mkdir_y_n_escape
3810 00008E2C 24DF <1> and al, 0DFh ; y -> Y, n -> N
3811 00008E2E 3C59 <1> cmp al, 'Y'; 'yes'
3812 00008E30 7404 <1> je short loc_mkdir_yes_make_directory
3813 00008E32 3C4E <1> cmp al, 'N'; 'no'
3814 00008E34 75EB <1> jne short loc_mkdir_ask_again
3815 <1>
3816 <1> loc_do_not_make_directory:
3817 <1> loc_mkdir_yes_make_directory:
3818 00008E36 E830000000 <1> call y_n_answer ; 29/12/2017
3819 <1> ;cmp al, 'Y'; 'yes'
3820 <1> ;cmc
3821 <1> ;jnc loc_file_rw_restore_retn
3822 00008E3B 3C4E <1> cmp al, 'N'; 'no'
3823 <1> ;je loc_file_rw_restore_retn
3824 <1> ; 28/07/2022

```

```

3825 00008E3D 7505      <1>     jne     short loc_mkdir_call_make_sub_dir
3826 00008E3F E9F4FEFFFF <1>     jmp     loc_file_rw_restore_retn
3827                                <1>
3828                                <1> loc_mkdir_call_make_sub_dir:
3829 00008E44 8B35[C0800100] <1>     mov     esi, [DelFile_FNPointer]
3830 00008E4A B110      <1>     mov     cl, 10h ; Directory attributes
3831 00008E4C E8001C0000 <1>     call    make_sub_directory
3832                                <1> loc_rename_file_ok: ; 06/03/2016
3833                                <1>     ;jc loc_file_rw_cmd_failed
3834                                <1>     ; 28/07/2022
3835 00008E51 7305      <1>     jnc     short move_source_file_to_dest_OK
3836 00008E53 E9E0FEFFFF <1>     jmp     loc_file_rw_cmd_failed
3837                                <1> move_source_file_to_dest_OK:
3838 00008E58 BE[18330100] <1>     mov     esi, Msg_OK
3839 00008E5D E848DFFFFF <1>     call    print_msg
3840 00008E62 E9D1FEFFFF <1>     jmp     loc_file_rw_restore_retn
3841                                <1>
3842                                <1> loc_mkdir_y_n_escape:
3843 00008E67 B04E      <1>     mov     al, 'N' ; 'no'
3844 00008E69 EBCB      <1>     jmp     short loc_do_not_make_directory
3845                                <1>
3846                                <1> y_n_answer:
3847                                <1>     ; 29/12/2017
3848 00008E6B A2[14330100] <1>     mov     [Y_N_nextline], al
3849                                <1>     ;push    ax
3850 00008E70 50          <1>     push    eax
3851 00008E71 BE[14330100] <1>     mov     esi, Y_N_nextline
3852 00008E76 E82FDFFFFF <1>     call    print_msg
3853 00008E7B 58          <1>     pop     eax
3854                                <1>     ;pop     ax
3855 00008E7C C3          <1>     retn
3856                                <1>
3857                                <1> delete_directory:
3858                                <1>     ; 17/07/2025 (TRDOS 386 Kernel v2.0.10)
3859                                <1>     ; 28/07/2022 (TRDOS 386 Kernel v2.0.5)
3860                                <1>     ; 29/12/2017
3861                                <1>     ; 15/10/2016
3862                                <1>     ; 01/03/2016, 06/03/2016
3863                                <1>     ; 27/02/2016, 28/02/2016, 29/02/2016
3864                                <1>     ; 26/02/2016 (TRDOS 386 = TRDOS v2.0)
3865                                <1>     ; 16/10/2010 (CMD_INTR.ASM, 'cmp_cmd_rmdir')
3866                                <1>     ; 05/06/2010
3867                                <1>
3868                                <1> get_fchar:
3869                                <1>     ; esi = directory name
3870 00008E7D 803E20 <1>     cmp     byte [esi], 20h
3871 00008E80 7701      <1>     ja      short loc_rmdir_parse_path_name
3872                                <1>
3873                                <1> loc_rmdir_nodirname_retn:
3874 00008E82 C3          <1>     retn
3875                                <1>
3876                                <1> loc_rmdir_parse_path_name:
3877 00008E83 BF[02800100] <1>     mov     edi, FindFile_Drv
3878 00008E88 E8E11A0000 <1>     call    parse_path_name
3879                                <1>     ;jc      loc_cmd_failed
3880                                <1>     ; 28/07/2022
3881 00008E8D 7305      <1>     jnc     short loc_rmdir_check_dirname_exists
3882                                <1> loc_del_dir_failed:
3883 00008E8F E9B0F5FFFF <1>     jmp     loc_cmd_failed
3884                                <1>
3885                                <1> loc_rmdir_check_dirname_exists:
3886 00008E94 BE[44800100] <1>     mov     esi, FindFile_Name
3887 00008E99 803E20 <1>     cmp     byte [esi], 20h
3888                                <1>     ;jna     loc_cmd_failed
3889                                <1>     ; 28/07/2022
3890 00008E9C 76F1      <1>     jna     short loc_del_dir_failed
3891 00008E9E 8935[C0800100] <1>     mov     [DelFile_FNPointer], esi
3892                                <1>
3893                                <1> loc_rmdir_drv:
3894 00008EA4 8A35[62770100] <1>     mov     dh, [Current_Drv]
3895 00008EAA 8835[BF7E0100] <1>     mov     [RUN_CDRV], dh
3896                                <1>
3897                                <1> ;;;;
3898                                <1> ; 17/07/2025 - TRDOS 386 v2.0.10
3899 00008EB0 8835[08810100] <1>     mov     [rmdir_drv], dh
3900 00008EB6 8B1D[5C770100] <1>     mov     ebx, [Current_Dir_FCluster]
3901 00008EBC 891D[04810100] <1>     mov     [rmdir_dir_fcluster], ebx
3902                                <1>     ;;;;
3903                                <1>
3904 00008EC2 8A15[02800100] <1>     mov     dl, [FindFile_Drv]
3905 00008EC8 38F2      <1>     cmp     dl, dh
3906 00008ECA 7407      <1>     je      short loc_rmdir_change_directory
3907                                <1>
3908 00008ECC E80FE8FFFF <1>     call    change_current_drive
3909                                <1>     ;jc      loc_file_rw_cmd_failed
3910                                <1>     ; 28/07/2022
3911                                <1>     ;jnc     short loc_rmdir_change_directory
3912                                <1>     ;jmp     loc_file_rw_cmd_failed
3913 00008ED1 7233      <1>     jc      short loc_rmdir_chdrv_failed
3914                                <1>
3915                                <1> loc_rmdir_change_directory:
3916 00008ED3 803D[03800100]20 <1>     cmp     byte [FindFile_Directory], 20h
3917 00008EDA 7614      <1>     jna     short loc_rmdir_find_directory
3918                                <1>
3919 00008EDC FE05[7E2E0100] <1>     inc     byte [Restore_CDIR]
3920 00008EE2 BE[03800100] <1>     mov     esi, FindFile_Directory
3921 00008EE7 30E4      <1>     xor     ah, ah ; CD_COMMAND sign -> 0
3922 00008EE9 E8ED140000 <1>     call    change_current_directory
3923 00008EEE 7211      <1>     jc      short loc_rmdir_check_error_code
3924                                <1>
3925                                <1> ;loc_rmdir_change_prompt_dir_string:
3926                                <1> ;call    change_prompt_dir_string
3927                                <1>
3928                                <1> loc_rmdir_find_directory:
3929                                <1>     ;mov     esi, FindFile_Name
3930 00008EF0 8B35[C0800100] <1>     mov     esi, [DelFile_FNPointer]
3931 00008EF6 66B81008 <1>     mov     ax, 0810h ; Only directories
3932 00008EFA E86BFAFFFF <1>     call    find_first_file
3933 00008EFF 730A      <1>     jnc     short loc_rmdir_ambgfn_check
3934                                <1>
3935                                <1> loc_rmdir_check_error_code:
3936 00008F01 3C02      <1>     cmp     al, 2
3937 00008F03 740B      <1>     je      short loc_rmdir_directory_not_found
3938 00008F05 F9          <1>     stc
3939                                <1> loc_rmdir_chdrv_failed:
3940 00008F06 E92DFEFFFF <1>     jmp     loc_file_rw_cmd_failed
3941                                <1>
3942                                <1> loc_rmdir_ambgfn_check:
3943 00008F0B 6621D2 <1>     and     dx, dx ; Ambiguous filename chars used sign (DX>0)
3944 00008F0E 740F      <1>     jz      short loc_rmdir_directory_found
3945                                <1>
3946                                <1> loc_rmdir_directory_not_found:
3947 00008F10 BE[89310100] <1>     mov     esi, Msg_Dir_Not_Found
3948 00008F15 E890DEFFFF <1>     call    print_msg

```

```

3949                                     <1>
3950 00008F1A E919FEFFFF               <1>      jmp      loc_file_rw_restore_retn
3951                                     <1>
3952 loc_rmdir_directory_found:          <1>
3953 00008F1F 80E307                   <1>      and      bl, 07h ; Attributes
3954                                     <1>      ;jnz      loc_permission_denied
3955                                     <1>      ; 28/07/2022
3956 00008F22 7405                     <1>      jz       short loc_rmdir_save_lnel
3957 00008F24 E919FEFFFF               <1>      jmp      loc_permission_denied
3958                                     <1>
3959 loc_rmdir_save_lnel: ; 28/02/2016   <1>
3960                                     <1>      ;mov bh, [LongName_EntryLength]
3961 00008F29 883D[CA800100]           <1>      mov      [DelFile_LNEL], bh ; Long name entry length (if > 0)
3962                                     <1>      ; edi = Directory Entry Offset (DirBuff)
3963                                     <1>      ; esi = Directory Entry (FFF Structure)
3964                                     <1>      ;mov      [DelFile_DirEntryAddr], edi ; not required
3965                                     <1>      ;mov      ax, [edi+20] ; First Cluster High word
3966                                     <1>      ;shl      eax, 16
3967                                     <1>      ;mov      ax, [edi+26] ; First Cluster Low word
3968                                     <1>      ; ROOT Dir First Cluster = 0
3969                                     <1>      ;cmp      eax, 2
3970                                     <1>      ;jb       loc_update_direntry_1
3971                                     <1>
3972 pass_rmdir_fc_check:                <1>
3973 00008F2F 57                       <1>      push     edi ; * (29/02/2016)
3974                                     <1>
3975 00008F30 BE[1E330100]             <1>      mov      esi, Msg_DoYouWantRmdir
3976 00008F35 E870DEFFFF               <1>      call     print_msg
3977 00008F3A 8B35[C0800100]           <1>      mov      esi, [DelFile_FNPointer]
3978 00008F40 E865DEFFFF               <1>      call     print_msg
3979 00008F45 BE[0A330100]             <1>      mov      esi, Msg_YesNo
3980 00008F4A E85BDEFFFF               <1>      call     print_msg
3981                                     <1>
3982 loc_rmdir_ask_again:                <1>
3983 00008F4F 30E4                     <1>      xor      ah, ah
3984 00008F51 E8C67FFFFF               <1>      call     int16h
3985 00008F56 3C1B                     <1>      cmp      al, 1Bh
3986                                     <1>      ;je       short loc_do_not_delete_directory
3987 00008F58 742F                     <1>      je       short loc_rmdir_y_n_escape ; 06/03/2016
3988 00008F5A 24DF                     <1>      and      al, 0DFh
3989 00008F5C A2[14330100]             <1>      mov      [Y_N_nextline], al
3990 00008F61 3C59                     <1>      cmp      al, 'Y'
3991 00008F63 7404                     <1>      je       short loc_rmdir_yes_delete_directory
3992 00008F65 3C4E                     <1>      cmp      al, 'N'
3993 00008F67 75E6                     <1>      jne      short loc_rmdir_ask_again
3994                                     <1>
3995 loc_do_not_delete_directory:         <1>
3996 loc_rmdir_yes_delete_directory:      <1>
3997 00008F69 E8FD7FFFFF               <1>      call     y_n_answer ; 29/12/2017
3998 00008F6E 5F                       <1>      pop      edi ; * (29/02/2016)
3999                                     <1>      ;cmp      al, 'Y' ; 'yes'
4000                                     <1>      ;cmc
4001                                     <1>      ;jnc      loc_file_rw_restore_retn
4002 00008F6F 3C4E                     <1>      cmp      al, 'N' ; 'no'
4003                                     <1>      ;je       loc_file_rw_restore_retn
4004                                     <1>      ; 28/07/2022
4005                                     <1>      ;jne      short loc_delete_sub_dir
4006                                     <1>      ;jmp      loc_file_rw_restore_retn
4007 00008F71 7411                     <1>      je       short loc_rmdir_rw_restore_retn
4008                                     <1>
4009 loc_delete_sub_dir:                 <1>
4010                                     <1>      ; 29/12/2017
4011 00008F73 E85E000000               <1>      call     delete_sub_directory
4012 00008F78 7213                     <1>      jc       short loc_rmdir_cmd_failed
4013                                     <1>
4014 loc_rmdir_ok:                       <1>
4015 00008F7A BE[18330100]             <1>      mov      esi, Msg_OK
4016 00008F7F E826DEFFFF               <1>      call     print_msg
4017 loc_rmdir_rw_restore_retn: ; 28/07/2022 <1>
4018 00008F84 E9AFFDFFFF               <1>      jmp      loc_file_rw_restore_retn
4019                                     <1>
4020 loc_rmdir_y_n_escape:                <1>
4021 00008F89 B04E                     <1>      mov      al, 'N' ; 'no'
4022 00008F8B EBDC                     <1>      jmp      loc_do_not_delete_directory
4023                                     <1>
4024 loc_rmdir_cmd_failed:                <1>
4025                                     <1>      ; 29/12/2017
4026 00008F8D 09C0                     <1>      or       eax, eax ; EAX = 0 -> Directory not empty!
4027 00008F8F 7423                     <1>      jz       short loc_rmdir_directory_not_empty
4028                                     <1>
4029                                     <1>      ; EAX > 0 -> Error code in AL (or AX or EAX)
4030                                     <1>
4031 00008F91 833D[7E7E0100]01         <1>      cmp      dword [FAT_ClusterCounter], 1
4032                                     <1>      ;jb       loc_file_rw_cmd_failed
4033                                     <1>      ; 28/07/2022
4034 00008F98 7305                     <1>      jnb      short loc_rmdir_failed
4035 00008F9A E999FDFFFF               <1>      jmp      loc_file_rw_cmd_failed
4036                                     <1>
4037 loc_rmdir_failed:                    <1>
4038 00008F9F F9                       <1>      stc
4039 loc_rmdir_cmd_return:                <1>
4040                                     <1>      ; 01/03/2016
4041 00008FA0 9C                       <1>      pushf
4042                                     <1>      ; ESI = Logical DOS Drive Description Table address
4043 00008FA1 66BB00FF                 <1>      mov      bx, 0FF00h ; BH = FFh -> use ESI for Drive parameters
4044                                     <1>      ; BL = 0 -> Recalculate free cluster count
4045 00008FA5 50                       <1>      push     eax
4046 00008FA6 E8B6350000               <1>      call     calculate_fat_freespace
4047 00008FAB 58                       <1>      pop      eax
4048 00008FAC 9D                       <1>      popf
4049                                     <1>      ;jc       loc_file_rw_cmd_failed
4050                                     <1>      ;jmp      loc_file_rw_restore_retn
4051                                     <1>      ; 28/07/2022
4052 00008FAD 73D5                     <1>      jnc      short loc_rmdir_rw_restore_retn
4053 00008FAF E984FDFFFF               <1>      jmp      loc_file_rw_cmd_failed
4054                                     <1>
4055 loc_rmdir_directory_not_empty:        <1>
4056 00008FB4 BE[3F330100]             <1>      mov      esi, Msg_Dir_Not_Empty
4057 00008FB9 E8ECDDFFFF               <1>      call     print_msg
4058                                     <1>      ; 01/03/2016
4059 00008FBE A1[7E7E0100]             <1>      mov      eax, [FAT_ClusterCounter]
4060 00008FC3 09C0                     <1>      or       eax, eax ; 0 ?
4061                                     <1>      ;jz       loc_file_rw_restore_retn
4062                                     <1>      ; 28/07/2022
4063 00008FC5 74BD                     <1>      jz       short loc_rmdir_rw_restore_retn
4064                                     <1>
4065                                     <1>      ; ESI = Logical DOS Drive Description Table address
4066 00008FC7 66BB01FF                 <1>      mov      bx, 0FF01h ; BH = FFh -> use ESI for Drive parameters
4067                                     <1>      ; BL = 1 -> add free clusters
4068 00008FCB E891350000               <1>      call     calculate_fat_freespace
4069 00008FD0 09C9                     <1>      or       ecx, ecx
4070                                     <1>      ;jz       loc_file_rw_restore_retn ; ecx = 0 -> OK
4071                                     <1>      ;; ecx > 0 -> Error (Recalculation is needed)
4072                                     <1>      ;jmp      short loc_rmdir_cmd_return

```

```

4073      <1>      ; 28/07/2022
4074 00008FD2 75CC      <1>      jnz      short loc_rmdir_cmd_return
4075 00008FD4 EBAA      <1>      jmp      short loc_rmdir_rw_restore_retn
4076      <1>      ; jmp      loc_file_rw_restore_retn
4077      <1>
4078      <1> delete_sub_directory:
4079      <1>      ; 19/12/2025
4080      <1>      ; 17/07/2025 (TRDOS 386 kernel v2.0.10)
4081      <1>      ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
4082      <1>      ; 29/12/2017
4083      <1>      ; (moved here from 'delete_directory' for 'sysrmdir' )
4084      <1>
4085      <1>      ; EDI = Directory buffer entry offset/address
4086      <1>
4087      <1> loc_rmdir_delete_short_name_check_dir_empty:
4088 00008FD6 668B4714      <1>      mov     ax, [edi+20] ; First Cluster High Word
4089 00008FDA C1E010      <1>      shl     eax, 16
4090 00008FDD 668B471A      <1>      mov     ax, [edi+26] ; First Cluster Low Word
4091      <1>
4092      <1>      ;;;
4093      <1>      ; 17/07/2025
4094      <1>      ; (sure it is not the current directory)
4095 00008FE1 8A15[08810100] <1>      mov     dl, [rmdir_drv]
4096 00008FE7 3A15[62770100] <1>      cmp     dl, [Current_Drv]
4097 00008FED 7508      <1>      jne     short loc_rmdir_delete_short_name_check_dot
4098 00008FEF 3B05[04810100] <1>      cmp     eax, [rmdir_dir_fcluster]
4099 00008FF5 743F      <1>      je      short loc_rmdir_permission_denied
4100      <1> loc_rmdir_delete_short_name_check_dot:
4101      <1>      ; (DOT and DOTDOT can not be deleted)
4102 00008FF7 803F2E      <1>      cmp     byte [edi], '.'
4103 00008FFA 750F      <1>      jne     short loc_rmdir_delete_short_name_skip_dot
4104 00008FFC 66813F2E20      <1>      cmp     word [edi], '..'
4105 00009001 7433      <1>      je      short loc_rmdir_permission_denied
4106 00009003 813F2E2E2020 <1>      cmp     dword [edi], '..'
4107 00009009 742B      <1>      je      short loc_rmdir_permission_denied
4108      <1> loc_rmdir_delete_short_name_skip_dot:
4109      <1>      ;;;
4110      <1>
4111      <1>      ; mov     [DelFile_FCluster], eax
4112      <1>
4113      <1>      ; mov     bx, [DirBuff_EntryCounter]
4114      <1>      ; mov     bx, [FindFile_DirEntryNumber] ; 27/02/2016
4115      <1>      ; mov     [DelFile_EntryCounter], bx
4116      <1>
4117 0000900B 29DB      <1>      sub     ebx, ebx
4118      <1>      ; 29/12/2017
4119 0000900D 891D[7E7E0100] <1>      mov     [FAT_ClusterCounter], ebx ; 0 ; Reset
4120      <1>
4121 00009013 8A3D[02800100] <1>      mov     bh, [FindFile_Drv]
4122 00009019 BE00010900      <1>      mov     esi, Logical_DOSDisks
4123 0000901E 01DE      <1>      add     esi, ebx
4124      <1>
4125      <1>      ; 19/12/2025
4126      <1>      ; cmp     word [edi+DirEntry_NTRes], 01A1h
4127      <1>      ; je      short loc_rmdir_check_fs_directory
4128      <1>
4129      <1>      ; cmp     byte [esi+LD_FATType], 1
4130      <1>      ; jb     short loc_rmdir_get_last_cluster_0
4131      <1>
4132      <1>      ; 29/12/2017
4133 00009020 83F802      <1>      cmp     eax, 2
4134 00009023 7306      <1>      jnb     short loc_rmdir_get_last_cluster_1
4135      <1>      ; eax < 2
4136      <1> loc_rmdir_get_last_cluster_0:
4137      <1>      ; mov     eax, ERR_INV_FORMAT ; invalid format!
4138 00009025 B813000000      <1>      mov     eax, ERR_NOT_DIR ; not a valid directory!
4139      <1>      ; stc
4140 0000902A C3      <1>      retn
4141      <1>
4142      <1> loc_rmdir_get_last_cluster_1:
4143 0000902B 807E0303      <1>      cmp     byte [esi+LD_FATType], 3 ; FAT32
4144 0000902F 750C      <1>      jne     short loc_rmdir_get_last_cluster_2
4145      <1>
4146      <1>      ; is it root directory ?
4147 00009031 3B4632      <1>      cmp     eax, [esi+LD_BPB+BPB_RootClus]
4148 00009034 7507      <1>      jne     short loc_rmdir_get_last_cluster_2
4149      <1>
4150      <1>      ; root directory can not be deleted !!
4151      <1> loc_rmdir_permission_denied:
4152 00009036 B80B000000      <1>      mov     eax, ERR_PERM_DENIED ; permission denied!
4153 0000903B F9      <1>      stc
4154 0000903C C3      <1>      retn
4155      <1>
4156      <1> loc_rmdir_get_last_cluster_2:
4157      <1>      ; 29/12/2017
4158 0000903D A3[C4800100] <1>      mov     [DelFile_FCluster], eax
4159      <1>
4160      <1>      ; mov     dx, [DirBuff_EntryCounter]
4161 00009042 668B15[7C800100] <1>      mov     dx, [FindFile_DirEntryNumber] ; 27/02/2016
4162 00009049 668915[C8800100] <1>      mov     [DelFile_EntryCounter], dx
4163      <1>
4164 00009050 8B15[8E7E0100] <1>      mov     edx, [DirBuff_Cluster]
4165 00009056 8915[F8800100] <1>      mov     [Rmdir_ParentDirCluster], edx
4166      <1>
4167 0000905C 893D[F4800100] <1>      mov     [Rmdir_DirEntryOffset], edi
4168      <1>
4169      <1>      ; 01/03/2016
4170      <1>      ; mov     dword [FAT_ClusterCounter], 0 ; Reset
4171      <1>
4172      <1> loc_rmdir_get_last_cluster_3:
4173 00009062 E878360000      <1>      call    get_last_cluster
4174      <1>      ; jc loc_rmdir_cmd_failed
4175 00009067 7211      <1>      jc      short loc_delete_sub_dir_retn ; 29/12/2017
4176      <1>
4177 00009069 3B05[C4800100] <1>      cmp     eax, [DelFile_FCluster]
4178 0000906F 750A      <1>      jne     short loc_rmdir_multi_dir_clusters
4179      <1>
4180 00009071 C605[F3800100]00 <1>      mov     byte [Rmdir_MultiClusters], 0
4181 00009078 EB08      <1>      jmp     short pass_rmdir_multi_dir_clusters
4182      <1>
4183      <1>      ; 19/12/2025
4184      <1>      %if 0
4185      <1>
4186      <1> loc_rmdir_check_fs_directory:
4187      <1>      ; 29/12/2017
4188      <1>      ; cmp     byte [esi+LD_FSType], 0A1h
4189      <1>      ; jne     short loc_rmdir_permission_denied
4190      <1>
4191      <1> loc_rmdir_delete_fs_directory:
4192      <1>      call    delete_fs_directory
4193      <1>      ; jnc     loc_print_deleted_message
4194      <1>      ; jnc     short loc_delete_sub_dir_retn ; 29/12/2017
4195      <1>
4196      <1>      ; EAX=0 -> Directory not empty !

```

```

4197      <1>      ; EAX>0 -> Disk r/w error or another (misc) error
4198      <1>
4199      <1>      ;or      eax, eax
4200      <1>      ;jz      loc_rmdir_directory_not_empty_2
4201      <1>      ;;stc
4202      <1>      ;;jmp     loc_file_rw_cmd_failed
4203      <1>
4204      <1> %endif
4205      <1>
4206      <1> loc_delete_sub_dir_retn:
4207      <1>      retn
4208      <1>
4209      <1> loc_rmdir_multi_dir_clusters:
4210      <1>      mov     byte [RmDir_MultiClusters], 1
4211      <1>
4212      <1> pass_rmdir_multi_dir_clusters:
4213      <1>      mov     [RmDir_DirLastCluster], eax
4214      <1>      mov     [RmDir_PreviousCluster], ecx
4215      <1>
4216      <1> loc_rmdir_load_fat_sub_directory:
4217      <1>      call    load_FAT_sub_directory
4218      <1>      ;jc      loc_rmdir_cmd_failed
4219      <1>      jc      short loc_delete_sub_dir_retn
4220      <1>
4221      <1> loc_rmdir_find_last_dir_entry:
4222      <1>      push    esi
4223      <1>      mov     esi, Dir_File_Name
4224      <1>      mov     byte [esi], '*'
4225      <1>      mov     byte [esi+8], '*'
4226      <1>      xor     ebx, ebx ; Entry offset = 0
4227      <1> loc_rmdir_find_last_dir_entry_next:
4228      <1>      ;mov     ax, 0800h ; Except volume/long names
4229      <1>      ;xor     cx, cx ; 0 = Find a valid file or dir name
4230      <1>      ; 28/07/2022
4231      <1>      xor     eax, eax
4232      <1>      mov     ah, 8
4233      <1>      ; eax = 0800h
4234      <1>      xor     ecx, ecx ; 0
4235      <1>      call    find_directory_entry
4236      <1>      jc      short loc_rmdir_empty_dir_cluster
4237      <1>      cmp     ebx, 1
4238      <1>      ja      short loc_rmdir_directory_not_empty_1
4239      <1> loc_rmdir_dot_entry_check:
4240      <1>      cmp     ch, '.' ; The first char of the dir entry
4241      <1>      jne     short loc_rmdir_directory_not_empty_1
4242      <1>      or      bl, bl
4243      <1>      jnz     short loc_rmdir_dotdot_entry_check
4244      <1>      cmp     byte [edi+1], 20h
4245      <1>      jmp     short pass_rmdir_dot_entry_check
4246      <1>
4247      <1> loc_rmdir_dotdot_entry_check:
4248      <1>      cmp     word [edi+1], '.'
4249      <1> pass_rmdir_dot_entry_check:
4250      <1>      jne     short loc_rmdir_directory_not_empty_1
4251      <1>      inc     bl
4252      <1>      jmp     short loc_rmdir_find_last_dir_entry_next
4253      <1>
4254      <1> loc_rmdir_directory_not_empty_1:
4255      <1>      pop     eax ; pushed esi
4256      <1>      xor     eax, eax ; 0
4257      <1> loc_rmdir_directory_not_empty_2:
4258      <1> loc_delete_sub_dir_stc_retn:
4259      <1>      stc
4260      <1>      retn
4261      <1>
4262      <1> loc_rmdir_empty_dir_cluster:
4263      <1>      pop     esi
4264      <1>
4265      <1> loc_rmdir_set_prev_cluster_dir_last_cluster:
4266      <1>      cmp     byte [RmDir_MultiClusters], 0
4267      <1>      jna     short loc_rmdir_unlink_dir_last_cluster
4268      <1>
4269      <1>      mov     eax, [RmDir_PreviousCluster]
4270      <1>      ;xor     ecx, ecx
4271      <1>      dec     ecx ; FFFFFFFFh
4272      <1>      call    update_cluster
4273      <1>      jnc     short loc_rmdir_unlink_dir_last_cluster
4274      <1>
4275      <1>      ; 01/03/2016
4276      <1>      ;cmp     eax, 1 ; eax = 0 -> end of cluster chain
4277      <1>      ;cmc
4278      <1>      ;jc      short loc_rmdir_cmd_failed
4279      <1>      ;jmp     short loc_rmdir_save_fat_buffer
4280      <1>      ; 29/12/2017
4281      <1>      and     eax, eax
4282      <1>      jnz     short loc_delete_sub_dir_stc_retn
4283      <1>      jmp     short loc_rmdir_save_fat_buffer
4284      <1>
4285      <1> loc_rmdir_unlink_dir_last_cluster:
4286      <1>      mov     eax, [RmDir_DirLastCluster]
4287      <1>      xor     ecx, ecx ; 0
4288      <1>      call    update_cluster
4289      <1>      jnc     short loc_rmdir_unlink_stc_retn_0Bh
4290      <1>      ; Because of it is the last cluster
4291      <1>      ; 'update_cluster' must return with eocc error
4292      <1>      or      eax, eax
4293      <1>      ;jz      short loc_rmdir_save_fat_buffer ; eocc
4294      <1>      ;stc
4295      <1>      ;jmp     short loc_rmdir_cmd_failed
4296      <1>      ; 29/12/2017
4297      <1>      jnz     short loc_delete_sub_dir_stc_retn
4298      <1>
4299      <1> loc_rmdir_save_fat_buffer:
4300      <1>      cmp     byte [FAT_BuffValidData], 2
4301      <1>      jne     short loc_rmdir_calculate_FAT_freespace
4302      <1>      call    save_fat_buffer
4303      <1>      ;jc      short loc_rmdir_cmd_failed
4304      <1>      ; 29/12/2017
4305      <1>      jc      short loc_rmdir_unlink_error_retn
4306      <1>
4307      <1>      ; 01/03/2016
4308      <1>      cmp     byte [RmDir_MultiClusters], 0
4309      <1>      jna     short loc_rmdir_calculate_FAT_freespace
4310      <1>
4311      <1>      mov     eax, [DelFile_FCluster]
4312      <1>      jmp     loc_rmdir_get_last_cluster_3
4313      <1>
4314      <1> loc_rmdir_unlink_stc_retn_0Bh:
4315      <1>      ; 15/10/2016 (0Bh -> 28)
4316      <1>      mov     eax, ERR_INV_FORMAT ; 28 = Invalid format
4317      <1> loc_rmdir_unlink_stc_retn:
4318      <1>      stc
4319      <1> loc_rmdir_unlink_error_retn:
4320      <1>      retn

```

```

4321 <1>
4322 <1> loc_rmdir_delete_short_name_invalid_data:
4323 <1> ;mov eax, 29 ; Invalid data (15/10/2016)
4324 <1> ; 28/07/2022
4325 0000912E 29C0 <1> sub eax, eax
4326 00009130 B01D <1> mov al, 29
4327 <1> ;stc
4328 <1> ;jmp loc_rmdir_cmd_failed
4329 <1> ; 29/12/2017
4330 00009132 EBF8 <1> jmp short loc_rmdir_unlink_stc_retn
4331 <1>
4332 <1> loc_rmdir_calculate_FAT_freespace:
4333 <1> ;mov eax, [FAT_ClusterCounter]
4334 <1> ; 29/12/2017
4335 00009134 29C0 <1> sub eax, eax ; 0
4336 00009136 8705[7E7E0100] <1> xchg eax, [FAT_ClusterCounter]
4337 <1> ;
4338 0000913C 66BB01FF <1> mov bx, 0FF01h
4339 <1> ; BL = 1 -> Add EAX to free space count
4340 <1> ; BH = FFh ->
4341 <1> ; ESI = Logical DOS Drive Description Table address
4342 00009140 E81C340000 <1> call calculate_fat_freespace
4343 <1>
4344 00009145 21C9 <1> and ecx, ecx ; ecx = 0 -> valid free sector count
4345 00009147 7409 <1> jz short loc_rmdir_delete_short_name_continue
4346 <1>
4347 <1> loc_rmdir_recalculate_FAT_freespace:
4348 00009149 66BB00FF <1> mov bx, 0FF00h ; BL = 0 -> Recalculate free space
4349 0000914D E80F340000 <1> call calculate_fat_freespace
4350 <1>
4351 <1> loc_rmdir_delete_short_name_continue:
4352 00009152 A1[F8800100] <1> mov eax, [RmDir_ParentDirCluster]
4353 00009157 83F802 <1> cmp eax, 2
4354 0000915A 7309 <1> jnb short loc_rmdir_del_short_name_load_sub_dir
4355 0000915C E8562F0000 <1> call load_FAT_root_directory
4356 <1> ;jc loc_file_rw_cmd_failed
4357 <1> ; 29/12/2017
4358 00009161 72CA <1> jc short loc_rmdir_unlink_error_retn
4359 00009163 EB07 <1> jmp short loc_rmdir_del_short_name_ld_chk_fclust
4360 <1>
4361 <1> loc_rmdir_del_short_name_load_sub_dir:
4362 00009165 E8CB2F0000 <1> call load_FAT_sub_directory
4363 <1> ;jc loc_file_rw_cmd_failed
4364 <1> ; 29/12/2017
4365 0000916A 72C1 <1> jc short loc_rmdir_unlink_error_retn
4366 <1>
4367 <1> loc_rmdir_del_short_name_ld_chk_fclust:
4368 0000916C 0FB73D[F4800100] <1> movzx edi, word [RmDir_DirEntryOffset]
4369 00009173 81C700000800 <1> add edi, Directory_Buffer
4370 <1>
4371 00009179 668B4714 <1> mov ax, [edi+20] ; First Cluster High word
4372 0000917D C1E010 <1> shl eax, 16
4373 00009180 668B471A <1> mov ax, [edi+26] ; First Cluster Low word
4374 <1> ; Not necessary...
4375 00009184 3B05[C4800100] <1> cmp eax, [DelFile_FCluster]
4376 0000918A 75A2 <1> jne short loc_rmdir_delete_short_name_invalid_data
4377 <1> ;
4378 0000918C C607E5 <1> mov byte [edi], 0E5h ; 'Deleted' sign
4379 <1> ; 27/02/2016
4380 <1> ; TRDOS v1 has a bug here! it does not set
4381 <1> ; 'DirBuff_ValidData' to 2; as result of this bug,
4382 <1> ; 'save_directory_buffer' would not save the change !
4383 0000918F C605[897E0100]02 <1> mov byte [DirBuff_ValidData], 2 ; change sign
4384 <1> ;
4385 00009196 E8E61B0000 <1> call save_directory_buffer
4386 <1> ;jc loc_file_rw_cmd_failed
4387 <1> ; 29/12/2017
4388 0000919B 7290 <1> jc short loc_rmdir_unlink_error_retn
4389 <1>
4390 <1> loc_rmdir_del_long_name:
4391 0000919D 0FB615[CA800100] <1> movzx edx, byte [DelFile_LNEL]
4392 000091A4 08D2 <1> or dl, dl
4393 000091A6 7410 <1> jz short loc_rmdir_update_parent_dir_lmdt
4394 <1>
4395 000091A8 0FB705[C8800100] <1> movzx eax, word [DelFile_EntryCounter]
4396 000091AF 29D0 <1> sub eax, edx
4397 <1> ; 29/12/2017
4398 000091B1 7205 <1> jc short loc_rmdir_update_parent_dir_lmdt
4399 <1>
4400 <1> ; EAX = Directory Entry Number of the long name last entry
4401 000091B3 E8091D0000 <1> call delete_longname
4402 <1>
4403 <1> loc_rmdir_update_parent_dir_lmdt:
4404 000091B8 E84D1C0000 <1> call update_parent_dir_lmdt
4405 <1> ;jc short loc_file_rw_cmd_failed
4406 <1> ; 29/12/2017
4407 <1> ;jc short loc_rmdir_unlink_error_retn
4408 <1>
4409 <1> loc_delete_sub_directory_ok:
4410 <1> ; 29/12/2017
4411 000091BD 31C0 <1> xor eax, eax ; 0 ; cf = 0
4412 000091BF C3 <1> retn
4413 <1>
4414 <1> delete_file:
4415 <1> ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
4416 <1> ; 29/02/2016
4417 <1> ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
4418 <1> ; 09/08/2010 (CMD_INTR.ASM, 'cmp_cmd_del')
4419 <1> ; 28/02/2010
4420 <1>
4421 <1> get_delfile_fchar:
4422 <1> ; esi = file name
4423 000091C0 803E20 <1> cmp byte [esi], 20h
4424 000091C3 7701 <1> ja short loc_delfile_parse_path_name
4425 <1>
4426 <1> loc_delfile_nofilename_retn:
4427 000091C5 C3 <1> retn
4428 <1>
4429 <1> loc_delfile_parse_path_name:
4430 000091C6 BF[02800100] <1> mov edi, FindFile_Drv
4431 000091CB E89E170000 <1> call parse_path_name
4432 <1> ;jc loc_cmd_failed
4433 <1> ; 28/07/2022
4434 000091D0 7305 <1> jnc short loc_delfile_check_filename_exists
4435 <1>
4436 000091D2 E96DF2FFFF <1> loc_delfile_failed: jmp loc_cmd_failed
4437 <1>
4438 <1> loc_delfile_check_filename_exists:
4439 000091D7 BE[44800100] <1> mov esi, FindFile_Name
4440 000091DC 803E20 <1> cmp byte [esi], 20h
4441 <1> ;jna loc_cmd_failed
4442 <1> ; 28/07/2022
4443 000091DF 76F1 <1> jna short loc_delfile_failed
4444 000091E1 8935[C0800100] <1> mov [DelFile_FNPointer], esi

```

```

4445 <1>
4446 <1> loc_delfile_drv:
4447 000091E7 8A15[02800100] <1> mov dl, [FindFile_Drv]
4448 000091ED 8A35[62770100] <1> mov dh, [Current_Drv]
4449 000091F3 8835[B7E0100] <1> mov [RUN_CDRV], dh
4450 000091F9 38F2 <1> cmp dl, dh
4451 000091FB 7407 <1> je short loc_delfile_change_directory
4452 <1>
4453 000091FD E8DEE4FFFF <1> call change_current_drive
4454 <1> ;jc loc_file_rw_cmd_failed
4455 <1> ; 28/07/2022
4456 <1> ;jnc short loc_delfile_change_directory
4457 <1> ;jmp loc_file_rw_cmd_failed
4458 00009202 721D <1> jc short loc_delfile_chdrv_failed
4459 <1>
4460 <1> loc_delfile_change_directory:
4461 00009204 803D[03800100]20 <1> cmp byte [FindFile_Directory], 20h
4462 0000920B 7619 <1> jna short loc_delfile_find
4463 <1>
4464 0000920D FE05[7E2E0100] <1> inc byte [Restore_CDIRE]
4465 00009213 BE[03800100] <1> mov esi, FindFile_Directory
4466 00009218 30E4 <1> xor ah, ah ; CD_COMMAND sign -> 0
4467 0000921A E8BC110000 <1> call change_current_directory
4468 <1> ;jc loc_file_rw_cmd_failed
4469 <1> ; 28/07/2022
4470 0000921F 7305 <1> jnc short loc_delfile_chdir_ok
4471 <1> loc_delfile_chdrv_failed:
4472 <1> loc_delfile_fff_failed:
4473 00009221 E912FBFFFF <1> jmp loc_file_rw_cmd_failed
4474 <1>
4475 <1> loc_delfile_chdir_ok: ; 28/07/2022
4476 <1>
4477 <1> ;loc_delfile_change_prompt_dir_string:
4478 <1> ;call change_prompt_dir_string
4479 <1>
4480 <1> loc_delfile_find:
4481 <1> ;mov esi, FindFile_Name
4482 00009226 8835[03800100] <1> mov esi, [DelFile_FNPointer]
4483 0000922C 66B80018 <1> mov ax, 1800h ; Except volume label and dirs
4484 00009230 E835F7FFFF <1> call find_first_file
4485 <1> ;jc loc_file_rw_cmd_failed
4486 <1> ; 28/07/2022
4487 00009235 72EA <1> jc short loc_delfile_fff_failed
4488 <1>
4489 <1> loc_delfile_ambgfn_check:
4490 00009237 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)
4491 0000923A 740A <1> jz short loc_delfile_found
4492 <1>
4493 <1> loc_file_not_found:
4494 <1> ;mov eax, 2 ; File not found sign
4495 <1> ; 28/07/2022
4496 0000923C 31C0 <1> xor eax, eax
4497 0000923E B002 <1> mov al, 2
4498 00009240 F9 <1> stc
4499 00009241 E9F2FAFFFF <1> jmp loc_file_rw_cmd_failed
4500 <1>
4501 <1> loc_delfile_found:
4502 00009246 80E307 <1> and bl, 07h ; Attributes
4503 <1> ;jnz loc_permission_denied
4504 <1> ; 28/07/2022
4505 00009249 7405 <1> jz short loc_delfile_attrb_ok
4506 0000924B E9F2FAFFFF <1> jmp loc_permission_denied
4507 <1>
4508 <1> loc_delfile_attrb_ok: ; 28/07/2022
4509 <1>
4510 <1> ;loc_delfile_found_save_lnel:
4511 <1> ; mov [DelFile_LNEL], bh ; Long name entry length (if > 0)
4512 <1>
4513 <1> loc_delfile_ask_for_delete:
4514 00009250 57 <1> push edi ; * (29/02/2016)
4515 <1>
4516 00009251 BE[56330100] <1> mov esi, Msg_DoYouWantDelete
4517 00009256 E84FDBFFFF <1> call print_msg
4518 0000925B 8835[C0800100] <1> mov esi, [DelFile_FNPointer]
4519 00009261 E844DBFFFF <1> call print_msg
4520 00009266 BE[0A330100] <1> mov esi, Msg_YesNo
4521 0000926B E83ADBFFFF <1> call print_msg
4522 <1>
4523 <1> loc_delfile_ask_again:
4524 00009270 30E4 <1> xor ah, ah
4525 00009272 E8A57CFFFF <1> call int16h
4526 00009277 3C1B <1> cmp al, 1Bh
4527 <1> ;je short loc_do_not_delete_file
4528 00009279 744C <1> je short loc_delfile_y_n_escape ; 06/03/2016
4529 0000927B 24DF <1> and al, 0DFh
4530 0000927D A2[14330100] <1> mov [Y_N_nextline], al
4531 00009282 3C59 <1> cmp al, 'Y'
4532 00009284 7404 <1> je short loc_yes_delete_file
4533 00009286 3C4E <1> cmp al, 'N'
4534 00009288 75E6 <1> jne short loc_delfile_ask_again
4535 <1>
4536 <1> loc_do_not_delete_file:
4537 <1> loc_yes_delete_file:
4538 0000928A E8DCFBFFFF <1> call y_n_answer ; 29/12/2017
4539 0000928F 5F <1> pop edi ; * (29/02/2016)
4540 <1> ;cmp al, 'Y' ; 'yes'
4541 <1> ;cmc
4542 <1> ;jnc loc_file_rw_restore_retn
4543 00009290 3C4E <1> cmp al, 'N' ; 'no'
4544 <1> ;je loc_file_rw_restore_retn
4545 <1> ; 28/07/2022
4546 00009292 7505 <1> jne short loc_delete_file
4547 00009294 E99FFAFFFF <1> jmp loc_file_rw_restore_retn
4548 <1>
4549 <1> loc_delete_file:
4550 00009299 8A3D[02800100] <1> mov bh, [FindFile_Drv]
4551 <1> ;mov bl, [DelFile_LNEL]
4552 0000929F 8A1D[51800100] <1> mov bl, [FindFile_LongNameEntryLength]
4553 <1> ;mov cx, [DirBuff_EntryCounter]
4554 000092A5 66B80D[7C800100] <1> mov cx, [FindFile_DirEntryNumber]
4555 <1> ; (*) EDI = Directory buffer entry offset/address
4556 000092AC E8E81D0000 <1> call remove_file ; (FILE.ASM, 'proc_delete_file')
4557 <1> ;jnc loc_print_deleted_message
4558 <1> ; 28/07/2022
4559 000092B1 7205 <1> jc short loc_delete_file_err1
4560 000092B3 E976FAFFFF <1> jmp loc_print_deleted_message
4561 <1>
4562 <1> loc_delete_file_err1:
4563 <1> ;cmp al, 05h
4564 000092B8 3C0B <1> cmp al, ERR_PERM_DENIED ; 29/12/2017 (5 -> 11)
4565 <1> ;je loc_permission_denied
4566 <1> ; 28/07/2022
4567 000092BA 7505 <1> jne short loc_delete_file_err2
4568 000092BC E981FAFFFF <1> jmp loc_permission_denied

```

```

4569      <1> loc_delete_file_err2:
4570 000092C1 F9      <1>      stc
4571 000092C2 E971FAFFFF <1>      jmp     loc_file_rw_cmd_failed
4572      <1>
4573      <1> loc_delfile_y_n_escape:
4574 000092C7 B04E      <1>      mov     al, 'N' ; 'no'
4575 000092C9 EBBF      <1>      jmp     short loc_do_not_delete_file
4576      <1>
4577      <1> set_file_attributes:
4578      <1>      ; 19/12/2025 (TRDOS 386 v2.0.10)
4579      <1>      ; 26/09/2024 (TRDOS 386 v2.0.9)
4580      <1>      ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
4581      <1>      ; 06/03/2016
4582      <1>      ; 04/03/2016 (TRDOS 386 = TRDOS v2.0)
4583      <1>      ; 10/07/2010 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_attr')
4584      <1>      ; 23/05/2010
4585      <1>      ; 17/12/2000 (P2000.ASM)
4586      <1>
4587      <1>      ; esi = file or directory name
4588      <1>      ;xor     ax, ax
4589      <1>      ; 28/07/2022
4590 000092CB 31C0      <1>      xor     eax, eax
4591 000092CD 66A3[A7330100] <1>      mov     [Attr_Chars], ax
4592 000092D3 A2[1C810100] <1>      mov     [Attributes], al
4593      <1>
4594      <1> get_attr_fchar:
4595      <1>      ; esi = file name
4596 000092D8 8A06      <1>      mov     al, [esi]
4597 000092DA 3C20      <1>      cmp     al, 20h
4598 000092DC 7621      <1>      jna     short loc_attr_file_nofilename_retn
4599      <1>
4600      <1> loc_scan_attr_params:
4601 000092DE 3C2D      <1>      cmp     al, '-'
4602      <1>      ;ja     loc_attr_file_parse_path_name
4603      <1>      ;je     short loc_attr_space
4604      <1>      ; 28/07/2022
4605 000092E0 7207      <1>      jnb     short loc_sfa_1
4606 000092E2 740E      <1>      je      short loc_attr_space
4607 000092E4 E911010000 <1>      jmp     loc_attr_file_parse_path_name
4608      <1>
4609      <1> loc_sfa_1:
4610      <1>      ; 28/07/2022
4611 000092E9 3C2B      <1>      cmp     al, '+'
4612      <1>      ;jne     loc_cmd_failed
4613 000092EB 7405      <1>      je      short loc_attr_space
4614      <1>
4615 000092ED E952F1FFFF <1>      jmp     loc_cmd_failed
4616      <1> loc_attr_space:
4617 000092F2 8A6601      <1>      mov     ah, [esi+1]
4618 000092F5 80FC20      <1>      cmp     ah, 20h
4619 000092F8 7706      <1>      ja      short pass_attr_space
4620      <1>      ;jnb     loc_cmd_failed
4621      <1>      ; 28/07/2022
4622 000092FA 72F1      <1>      jnb     short loc_sfa_2
4623 000092FC 46      <1>      inc     esi
4624 000092FD EBF3      <1>      jmp     short loc_attr_space
4625      <1>
4626      <1> loc_attr_file_nofilename_retn:
4627 000092FF C3      <1>      retn
4628      <1>
4629      <1> pass_attr_space:
4630 00009300 80E4DF      <1>      and     ah, 0DFh
4631 00009303 80FC53      <1>      cmp     ah, 'S'
4632      <1>      ;ja     loc_cmd_failed
4633      <1>      ; 28/07/2022
4634 00009306 77E5      <1>      ja      short loc_sfa_2
4635 00009308 7204      <1>      jnb     short pass_attr_system
4636 0000930A B404      <1>      mov     ah, 04h ; System
4637 0000930C EB1D      <1>      jmp     short pass_attr_archive
4638      <1>
4639      <1> pass_attr_system:
4640 0000930E 80FC48      <1>      cmp     ah, 'H'
4641 00009311 7706      <1>      ja      short pass_attr_hidden
4642 00009313 720F      <1>      jnb     short pass_attr_read_only
4643 00009315 B402      <1>      mov     ah, 02h ; Hidden
4644 00009317 EB12      <1>      jmp     short pass_attr_archive
4645      <1>
4646      <1> pass_attr_hidden:
4647 00009319 80FC52      <1>      cmp     ah, 'R'
4648      <1>      ;ja     loc_cmd_failed
4649      <1>      ; 28/07/2022
4650 0000931C 77CF      <1>      ja      short loc_sfa_2
4651 0000931E 7204      <1>      jnb     short pass_attr_read_only ; Read only
4652 00009320 B401      <1>      mov     ah, 01h
4653 00009322 EB07      <1>      jmp     short pass_attr_archive
4654      <1>
4655      <1> pass_attr_read_only:
4656 00009324 80FC41      <1>      cmp     ah, 'A'
4657 00009327 753B      <1>      jne     short loc_chk_attr_enter
4658 00009329 B420      <1>      mov     ah, 20h ; Archive
4659      <1>
4660      <1> pass_attr_archive:
4661 0000932B 3C2D      <1>      cmp     al, '-'
4662 0000932D 7508      <1>      jne     short pass_reducing_attributes
4663 0000932F 0825[A7330100] <1>      or      [Attr_Chars], ah
4664 00009335 EB06      <1>      jmp     short loc_change_attributes_inc
4665      <1>
4666      <1> pass_reducing_attributes:
4667 00009337 0825[A8330100] <1>      or      [Attr_Chars+1], ah
4668      <1>
4669      <1> loc_change_attributes_inc:
4670 0000933D 46      <1>      inc     esi
4671 0000933E 8A6601      <1>      mov     ah, [esi+1]
4672 00009341 80FC20      <1>      cmp     ah, 20h
4673 00009344 7228      <1>      jnb     short pass_change_attr
4674 00009346 74F5      <1>      je      short loc_change_attributes_inc
4675 00009348 80FC2D      <1>      cmp     ah, '-'
4676 0000934B 770D      <1>      ja      short loc_chk_next_attr_char1
4677 0000934D 7405      <1>      je      short loc_chk_next_attr_char0
4678 0000934F 80FC2B      <1>      cmp     ah, '+'
4679 00009352 7506      <1>      jne     short loc_chk_next_attr_char1
4680      <1>
4681      <1> loc_chk_next_attr_char0:
4682 00009354 46      <1>      inc     esi
4683 00009355 668B06      <1>      mov     ax, [esi]
4684 00009358 EBA6      <1>      jmp     short pass_attr_space
4685      <1>
4686      <1> loc_chk_next_attr_char1:
4687 0000935A 803E2D      <1>      cmp     byte [esi], '-'
4688 0000935D 77A1      <1>      ja      short pass_attr_space
4689 0000935F E989000000 <1>      jmp     loc_attr_file_check_fname_fchar
4690      <1>
4691      <1> loc_chk_attr_enter:
4692 00009364 80FC0D      <1>      cmp     ah, 0Dh

```



```

4693             <1>     ;jne     loc_cmd_failed
4694             <1>     ; 28/07/2022
4695 00009367 7405     <1>     je      short pass_change_attr
4696 00009369 E9D6F0FFFF <1>     jmp     loc_cmd_failed
4697             <1>
4698             <1> pass_change_attr:
4699 0000936E A0[A7330100] <1>     mov     al, [Attr_Chars]
4700 00009373 F6D0     <1>     not     al
4701 00009375 2005[1C810100] <1>     and     [Attributes], al
4702 0000937B A0[A8330100] <1>     mov     al, [Attr_Chars+1]
4703 00009380 0805[1C810100] <1>     or      [Attributes], al
4704             <1>
4705             <1> loc_show_attributes:
4706 00009386 BE[07390100] <1>     mov     esi, nextline
4707 0000938B E81ADAF0FF <1>     call    print_msg
4708             <1>
4709             <1> loc_show_attributes_no_nextline:
4710 00009390 C705[A7330100]4E4F- <1>     mov     dword [Attr_Chars], 'NORM'
4711 00009398 524D     <1>
4712 0000939A 66C705[AB330100]41- <1>     mov     word [Attr_Chars+4], 'AL'
4713 000093A2 4C      <1>
4714 000093A3 BE[A7330100] <1>     mov     esi, Attr_Chars
4715 000093A8 A0[1C810100] <1>     mov     al, [Attributes]
4716 000093AD A804     <1>     test    al, 04h
4717 000093AF 7406     <1>     jz      short pass_put_attr_s
4718 000093B1 66C7065300 <1>     mov     word [esi], 0053h ; S
4719 000093B6 46      <1>     inc     esi
4720             <1>
4721             <1> pass_put_attr_s:
4722 000093B7 A802     <1>     test    al, 02h
4723 000093B9 7406     <1>     jz      short pass_put_attr_h
4724 000093BB 66C7064800 <1>     mov     word [esi], 0048h ; H
4725 000093C0 46      <1>     inc     esi
4726             <1>
4727             <1> pass_put_attr_h:
4728 000093C1 A801     <1>     test    al, 01h
4729 000093C3 7406     <1>     jz      short pass_put_attr_r
4730 000093C5 66C7065200 <1>     mov     word [esi], 0052h ; R
4731 000093CA 46      <1>     inc     esi
4732             <1>
4733             <1> pass_put_attr_r:
4734 000093CB 3C20     <1>     cmp     al, 20h
4735 000093CD 7205     <1>     jnb     short pass_put_attr_a
4736 000093CF 66C7064100 <1>     mov     word [esi], 0041h ; A
4737             <1>
4738             <1> pass_put_attr_a:
4739 000093D4 BE[9A330100] <1>     mov     esi, Str_Attributes
4740 000093D9 E8CCD9FFFF <1>     call    print_msg
4741 000093DE BE[07390100] <1>     mov     esi, nextline
4742 000093E3 E8C2D9FFFF <1>     call    print_msg
4743 000093E8 E94BF9FFFF <1>     jmp     loc_file_rw_restore_retn
4744             <1>
4745             <1> loc_attr_file_check_fname_fchar:
4746 000093ED 46      <1>     inc     esi
4747 000093EE 803E20 <1>     cmp     byte [esi], 20h
4748 000093F1 74FA     <1>     je      short loc_attr_file_check_fname_fchar
4749             <1> ;jb pass_change_attr
4750             <1> ; 28/07/2022
4751             <1> ja      short loc_attr_file_parse_path_name
4752             <1> jmp     pass_change_attr
4753             <1>
4754             <1> loc_attr_file_parse_path_name:
4755 000093FA BF[02800100] <1>     mov     edi, FindFile_Drv
4756 000093FF E86A150000 <1>     call    parse_path_name
4757             <1> ;jc      loc_cmd_failed
4758             <1> ; 28/07/2022
4759             <1> jnc     short loc_attr_file_check_filename_exists
4760             <1>
4761             <1> loc_sfa_3:
4762 00009406 E939F0FFFF <1>     jmp     loc_cmd_failed
4763             <1>
4764             <1> loc_attr_file_check_filename_exists:
4765 0000940B BE[44800100] <1>     mov     esi, FindFile_Name
4766 00009410 803E20 <1>     cmp     byte [esi], 20h
4767             <1> ;jna     loc_cmd_failed
4768             <1> ; 28/07/2022
4769             <1> jna     short loc_sfa_3
4770             <1> mov     [DelFile_FNPointer], esi
4771             <1>
4772             <1> loc_attr_file_drv:
4773 0000941B 8A35[62770100] <1>     mov     dh, [Current_Drv]
4774 00009421 8835[BF7E0100] <1>     mov     [RUN_CDRV], dh
4775             <1>
4776             <1>
4777 00009427 8A15[02800100] <1>     mov     dl, [FindFile_Drv]
4778 0000942D 38F2     <1>     cmp     dl, dh
4779 0000942F 7407     <1>     je      short loc_attr_file_change_directory
4780             <1>
4781             <1>
4782             <1> call    change_current_drive
4783             <1> ;jc      loc_file_rw_cmd_failed
4784             <1> ; 28/07/2022
4785             <1> jc      short loc_sfa_4
4786             <1>
4787             <1> loc_attr_file_change_directory:
4788 00009438 803D[03800100]20 <1>     cmp     byte [FindFile_Directory], 20h
4789 0000943F 7614     <1>     jna     short loc_attr_file_find
4790             <1>
4791             <1> inc     byte [Restore_CDIR]
4792             <1>
4793             <1> mov     esi, FindFile_Directory
4794             <1> xor     ah, ah ; CD_COMMAND sign -> 0
4795             <1> call    change_current_directory
4796             <1> ;jc      loc_file_rw_cmd_failed
4797             <1> ; 28/07/2022
4798             <1> jc      short loc_sfa_4
4799             <1>
4800             <1> ;loc_attr_file_change_prompt_dir_string:
4801             <1> ;call    change_prompt_dir_string
4802             <1>
4803             <1> loc_attr_file_find:
4804             <1> ;mov     esi, FindFile_Name
4805             <1> mov     esi, [DelFile_FNPointer]
4806             <1> mov     ax, 0800h ; Except volume labels
4807             <1> call    find_first_file
4808             <1> ;jc      loc_file_rw_cmd_failed
4809             <1> ; 28/07/2022
4810             <1> jnc     short loc_attr_file_ambgfn_check
4811             <1>
4812             <1> loc_sfa_4:
4813             <1> jmp     loc_file_rw_cmd_failed
4814             <1>
4815             <1> loc_attr_file_ambgfn_check:
4816             <1> or      dx, dx ; Ambiguous filename chars used sign (DX>0)
4817             <1> ; (Note: It was BX in TRDOS v1)
4818             <1> ;jz      short loc_attr_file_found
4819             <1> ;jnz     loc_file_not_found ; 06/03/2016
4820             <1> ; 28/07/2022

```

```

4815 0000946E 7405      <1>      jz      short loc_attr_file_found
4816 00009470 E9C7FDFFFF <1>      jmp     loc_file_not_found
4817                                <1>
4818                                <1>      ;mov     eax, 2 ; File not found sign
4819                                <1>      ;stc
4820                                <1>      ;jmp     loc_file_rw_cmd_failed
4821                                <1>
4822                                <1> loc_attr_file_found:
4823                                <1>      ; EDI = Directory buffer entry offset/address
4824                                <1>      ; BL = File (or Directory) Attributes
4825                                <1>      ; (Note: It was 'CL' in TRDOS v1)
4826                                <1>      ; mov     bl, [EDI+0Bh]
4827                                <1>
4828 00009475 66833D[A7330100]00 <1>      cmp     word [Attr_Chars], 0
4829 0000947D 770B      <1>      ja      short loc_attr_file_change_attributes
4830 0000947F 881D[1C810100] <1>      mov     [Attributes], bl
4831 00009485 E9FCFEFFFF <1>      jmp     loc_show_attributes
4832                                <1>
4833                                <1> loc_attr_file_change_attributes:
4834 0000948A A0[A7330100] <1>      mov     al, [Attr_Chars]
4835 0000948F F6D0      <1>      not     al
4836 00009491 20C3      <1>      and     bl, al
4837 00009493 A0[A8330100] <1>      mov     al, [Attr_Chars+1]
4838 00009498 08C3      <1>      or      bl, al
4839                                <1>
4840                                <1>      ; 19/12/2025
4841                                <1>      ;cmp     word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
4842                                <1>      ;je      short loc_attr_file_fs_check
4843                                <1>
4844 0000949A 881D[1C810100] <1>      mov     [Attributes], bl
4845 000094A0 885F0B      <1>      mov     [edi+0Bh], bl ; Attributes (New!)
4846                                <1>
4847                                <1>      ; 04/03/2016
4848                                <1>      ; TRDOS v1 has a bug here! it does not set
4849                                <1>      ; 'DirBuff_ValidData' to 2; as result of this bug,
4850                                <1>      ; 'save_directory_buffer' would not save the new attributes !
4851                                <1>
4852 000094A3 C605[897E0100]02 <1>      mov     byte [DirBuff_ValidData], 2
4853                                <1>
4854 000094AA E8D2180000 <1>      call    save_directory_buffer
4855                                <1>      ;jc      loc_file_rw_cmd_failed
4856                                <1>      ;jmp     short loc_print_attr_changed_message
4857                                <1>      ; 28/07/2022
4858 000094AF 7305      <1>      jnc     short loc_print_attr_changed_message
4859                                <1> loc_sfa_5:
4860 000094B1 E982F8FFFF <1>      jmp     loc_file_rw_cmd_failed
4861                                <1>
4862                                <1>      ; 19/12/2025
4863                                <1>      %if 0
4864                                <1>      loc_attr_file_fs_check:
4865                                <1>      sub     eax, eax
4866                                <1>      mov     ah, [DirBuff_DRV]
4867                                <1>      ; 26/09/2024 (BugFix)
4868                                <1>      sub     ah, 'A'
4869                                <1>      mov     esi, Logical_DOSDisks
4870                                <1>      add     esi, eax
4871                                <1>      cmp     byte [esi+LD_FSType], 0A1h
4872                                <1>      jnc     short loc_attr_file_change_fs_file_attributes
4873                                <1>      ; 29/12/2017 (0Dh -> 29)
4874                                <1>      mov     ax, 29 ; Invalid Data
4875                                <1>      jmp     loc_file_rw_cmd_failed
4876                                <1>
4877                                <1> loc_attr_file_change_fs_file_attributes:
4878                                <1>      ; BL = New MS-DOS File Attributes
4879                                <1>      mov     al, bl ; File/Directory Attributes
4880                                <1>      xor     ah, ah ; Attributes in MS-DOS format sign
4881                                <1>      call    change_fs_file_attributes
4882                                <1>      ;jc      loc_file_rw_cmd_failed
4883                                <1>      ; 28/07/2022
4884                                <1>      jc      short loc_sfa_5
4885                                <1>
4886                                <1>      mov     [Attributes], bl
4887                                <1>      %endif
4888                                <1>
4889                                <1> loc_print_attr_changed_message:
4890 000094B6 BE[95330100] <1>      mov     esi, Msg_New
4891 000094BB E8EAD8FFFF <1>      call    print_msg
4892 000094C0 E9CBFEFFFF <1>      jmp     loc_show_attributes_no_nextline
4893                                <1>
4894                                <1> rename_file:
4895                                <1>      ; 25/07/2025 (BugFix)
4896                                <1>      ; 28/07/2022 (TRDOS 386 Kernel v2.0.5)
4897                                <1>      ; 13/11/2017
4898                                <1>      ; 06/11/2016
4899                                <1>      ; 05/11/2016
4900                                <1>      ; 16/10/2016
4901                                <1>      ; 08/03/2016
4902                                <1>      ; 06/03/2016 (TRDOS 386 = TRDOS v2.0)
4903                                <1>      ; 20/11/2010 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_rename')
4904                                <1>      ; 16/11/2010
4905                                <1>
4906                                <1> get_rename_source_fchar:
4907                                <1>      ; esi = file name
4908 000094C5 803E20 <1>      cmp     byte [esi], 20h
4909 000094C8 7613      <1>      jna     short loc_rename_nofilename_retn
4910                                <1>
4911 000094CA 8935[44810100] <1>      mov     [SourceFilePath], esi
4912                                <1>
4913                                <1> rename_scan_source_file:
4914 000094D0 46      <1>      inc     esi
4915 000094D1 803E20 <1>      cmp     byte [esi], 20h
4916 000094D4 7408      <1>      je      short rename_scan_destination_file_1
4917                                <1>      ;jnb     short loc_rename_nofilename_retn
4918                                <1>      ;jb      loc_cmd_failed
4919                                <1>      ;jmp     short rename_scan_source_file
4920                                <1>      ; 28/07/2022
4921 000094D6 77F8      <1>      ja      short rename_scan_source_file
4922                                <1> loc_rename_failed:
4923 000094D8 E967EFFFFF <1>      jmp     loc_cmd_failed
4924                                <1>
4925                                <1> loc_rename_nofilename_retn: ; 08/03/2016
4926 000094DD C3      <1>      retn
4927                                <1>
4928                                <1> rename_scan_destination_file_1:
4929 000094DE C60600 <1>      mov     byte [esi], 0
4930                                <1>
4931                                <1> rename_scan_destination_file_2:
4932 000094E1 46      <1>      inc     esi
4933 000094E2 803E20 <1>      cmp     byte [esi], 20h
4934 000094E5 74FA      <1>      je      short rename_scan_destination_file_2
4935                                <1>      ;jnb     short loc_rename_nofilename_retn
4936                                <1>      ;jb      loc_cmd_failed
4937                                <1>      ; 28/07/2022
4938 000094E7 72EF      <1>      jb      short loc_rename_failed

```

```

4939                                     <1>
4940 000094E9 8935[48810100]          <1>      mov     [DestinationFilePath], esi
4941                                     <1>
4942                                     <1> rename_scan_destination_file_3:
4943 000094EF 46                        <1>      inc     esi
4944 000094F0 803E20                  <1>      cmp     byte [esi], 20h
4945 000094F3 77FA                    <1>      ja      short rename_scan_destination_file_3
4946                                     <1>
4947 000094F5 C60600                  <1>      mov     byte [esi], 0
4948                                     <1>
4949                                     <1> loc_rename_save_current_drive:
4950 000094F8 8A35[62770100]          <1>      mov     dh, [Current_Drv]
4951 000094FE 8835[B7E0100]          <1>      mov     byte [RUN_CDRV], dh
4952                                     <1>
4953                                     <1> loc_rename_sf_parse_path_name:
4954 00009504 8B35[44810100]          <1>      mov     esi, [SourceFilePath]
4955 0000950A BF[02800100]          <1>      mov     edi, FindFile_Drv
4956 0000950F E85A140000              <1>      call    parse_path_name
4957                                     <1>      ;jc      loc_cmd_failed
4958                                     <1>      ; 28/07/2022
4959 00009514 72C2                    <1>      jc      short loc_rename_failed
4960                                     <1>
4961                                     <1> loc_rename_sf_check_filename_exists:
4962 00009516 BE[44800100]          <1>      mov     esi, FindFile_Name
4963 0000951B 803E20                  <1>      cmp     byte [esi], 20h
4964                                     <1>      ;jna     loc_cmd_failed
4965                                     <1>      ; 28/07/2022
4966 0000951E 76B8                    <1>      jna     short loc_rename_failed
4967                                     <1>
4968                                     <1>      ;mov     [DelFile_FNPointer], esi
4969                                     <1>
4970                                     <1> loc_rename_sf_drv:
4971                                     <1>      ;mov     dh, [Current_Drv]
4972                                     <1>      ;mov     [RUN_CDRV], dh
4973                                     <1>
4974 00009520 8A15[02800100]          <1>      mov     dl, [FindFile_Drv]
4975 00009526 38F2                    <1>      cmp     dl, dh ; dh = [Current_Drv]
4976 00009528 7407                    <1>      je      short rename_sf_change_directory
4977                                     <1>
4978 0000952A E8B1E1FFFF              <1>      call    change_current_drive
4979                                     <1>      ;jc      loc_file_rw_cmd_failed
4980                                     <1>      ; 28/07/2022
4981 0000952F 722D                    <1>      jc      short loc_rename_fff_failed
4982                                     <1>
4983                                     <1> rename_sf_change_directory:
4984 00009531 803D[03800100]20        <1>      cmp     byte [FindFile_Directory], 20h
4985 00009538 7614                    <1>      jna     short rename_sf_find
4986                                     <1>
4987 0000953A FE05[7E2E0100]          <1>      inc     byte [Restore_CDIR]
4988 00009540 BE[03800100]          <1>      mov     esi, FindFile_Directory
4989 00009545 30E4                    <1>      xor     ah, ah ; CD_COMMAND sign -> 0
4990 00009547 E88F0E0000              <1>      call    change_current_directory
4991                                     <1>      ;jc      loc_file_rw_cmd_failed
4992                                     <1>      ; 28/07/2022
4993 0000954C 7210                    <1>      jc      short loc_rename_fff_failed
4994                                     <1>
4995                                     <1> ;rename_sf_change_prompt_dir_string:
4996                                     <1>      ;call    change_prompt_dir_string
4997                                     <1>
4998                                     <1> rename_sf_find:
4999                                     <1>      ;mov     esi, [DelFile_FNPointer]
5000 0000954E BE[44800100]          <1>      mov     esi, FindFile_Name
5001                                     <1>
5002 00009553 66B80008                <1>      mov     ax, 0800h ; Except volume labels
5003 00009557 E80EF4FFFF              <1>      call    find_first_file
5004                                     <1>      ;jc      loc_file_rw_cmd_failed
5005                                     <1>      ; 28/07/2022
5006 0000955C 7305                    <1>      jnc     short loc_rename_sf_ambgfn_check
5007                                     <1>
5008                                     <1> loc_rename_fff_failed:
5009 0000955E E9D5F7FFFF              <1>      jmp     loc_file_rw_cmd_failed
5010                                     <1>
5011                                     <1> loc_rename_sf_ambgfn_check:
5012 00009563 6621D2                  <1>      and     dx, dx ; Ambiguous filename chars used sign (DX>0)
5013                                     <1>      ; (Note: It was BX in TRDOS v1)
5014                                     <1>      ;jz      short loc_rename_sf_found
5015                                     <1>      ;jnz     loc_file_not_found
5016                                     <1>      ; 28/07/2022
5017 00009566 7405                    <1>      jz      short loc_rename_sf_found
5018 00009568 E9CFFCFFFF              <1>      jmp     loc_file_not_found
5019                                     <1>
5020                                     <1>      ;mov     eax, 2 ; File not found sign
5021                                     <1>      ;stc
5022                                     <1>      ;jmp     loc_file_rw_cmd_failed
5023                                     <1>
5024                                     <1> loc_rename_sf_found:
5025                                     <1>      ; EDI = Directory buffer entry offset/address
5026                                     <1>      ; BL = File (or Directory) Attributes
5027                                     <1>      ; (Note: It was 'CL' in TRDOS v1)
5028                                     <1>      ; mov     bl, [EDI+0Bh]
5029                                     <1>
5030 0000956D F6C307                  <1>      test    bl, 07h ; Attributes, S-H-R
5031                                     <1>      ;jnz     loc_permission_denied
5032                                     <1>      ; 28/07/2022
5033 00009570 7405                    <1>      jz      short loc_rename_attrb_ok
5034 00009572 E9CBF7FFFF              <1>      jmp     loc_permission_denied
5035                                     <1>
5036                                     <1> loc_rename_attrb_ok:
5037 00009577 BE[02800100]          <1>      mov     esi, FindFile_Drv
5038 0000957C BF[4C810100]          <1>      mov     edi, SourceFile_Drv
5039 00009581 B920000000              <1>      mov     ecx, 32
5040 00009586 F3A5                    <1>      rep     movsd
5041                                     <1>
5042                                     <1> loc_rename_df_parse_path_name:
5043 00009588 8B35[48810100]          <1>      mov     esi, [DestinationFilePath]
5044 0000958E BF[02800100]          <1>      mov     edi, FindFile_Drv
5045 00009593 E8D6130000              <1>      call    parse_path_name
5046 00009598 7219                    <1>      jc      short loc_rename_df_cmd_failed
5047                                     <1>
5048                                     <1>      ;mov     dh, [RUN_CDRV]
5049 0000959A 8A35[62770100]          <1>      mov     dh, [Current_Drv]
5050                                     <1>
5051                                     <1>      ; 'rename' command is valid only for same dos drive and same dir!
5052                                     <1>      ; ('move' command must be used if source file and destination file
5053                                     <1>      ; directories are not same!)
5054 000095A0 8A15[02800100]          <1>      mov     dl, [FindFile_Drv]
5055 000095A6 38F2                    <1>      cmp     dl, dh ; are source and destination drives different ?!
5056 000095A8 7509                    <1>      jne     short loc_rename_df_cmd_failed ; yes!
5057                                     <1>
5058                                     <1> rename_df_check_dirname_exists:
5059 000095AA 803D[03800100]00        <1>      cmp     byte [FindFile_Directory], 0
5060 000095B1 760B                    <1>      jna     short rename_df_check_filename_exists
5061                                     <1>
5062                                     <1>      ; different source file and destination file directories !

```

```

5063                                     <1> loc_rename_df_cmd_failed:
5064 000095B3 B801000000                 <1>     mov     eax, 1 ; TRDOS 'Bad command or file name' error
5065 000095B8 F9                       <1>     stc
5066 000095B9 E97AF7FFFF                 <1>     jmp     loc_file_rw_cmd_failed
5067                                     <1>
5068                                     <1> rename_df_check_filename_exists:
5069 000095BE BE[44800100]               <1>     mov     esi, FindFile_Name
5070 000095C3 E801F7FFFF                 <1>     call    check_filename
5071                                     <1>     ;jc      loc_mkdir_invalid_dir_name_chars
5072                                     <1>     ; 28/07/2022
5073 000095C8 7305                       <1>     jnc     short loc_rename_file_name_ok
5074 000095CA E91BF8FFFF                 <1>     jmp     loc_mkdir_invalid_dir_name_chars
5075                                     <1>
5076                                     <1> loc_rename_file_name_ok:
5077                                     <1>     ;mov     [DelFile_FNPointer], esi
5078                                     <1>     ;cmp     byte [esi], 20h
5079                                     <1>     ;ja      short loc_rename_df_find
5080                                     <1>
5081                                     <1>     ;mov     dh, [Current_Drv] ; dh has not been changed
5082                                     <1>
5083                                     <1> rename_df_drv_check_writable:
5084                                     <1>     ;movzx   esi, dh
5085                                     <1>     ;movzx   esi, byte [Current_Drv]
5086                                     <1>     ;add     esi, Logical_DOSDisks
5087                                     <1>     ; 25/07/2025 (BugFix)
5088 000095CF 31C0                       <1>     xor     eax, eax
5089 000095D1 88F4                       <1>     mov     ah, dh
5090 000095D3 BE00010900                 <1>     mov     esi, Logical_DOSDisks
5091 000095D8 01C6                       <1>     add     esi, eax
5092                                     <1>
5093 000095DA 88F2                       <1>     mov     dl, dh ; dl = [Current_Drv]
5094 000095DC 8A7601                     <1>     mov     dh, [esi+LD_DiskType]
5095                                     <1>
5096 000095DF 80FE01                     <1>     cmp     dh, 1 ; 0 = Invalid
5097 000095E2 7310                       <1>     jnb     short rename_df_compare_sf_df_name
5098                                     <1>
5099                                     <1>     ; 16/10/2016 (13h -> 30)
5100 000095E4 B81E000000                 <1>     mov     eax, 30 ; 'Disk write-protected' error
5101 000095E9 8B1D[48810100]             <1>     mov     ebx, [DestinationFilePath]
5102 000095EF E944F7FFFF                 <1>     jmp     loc_file_rw_cmd_failed
5103                                     <1>
5104                                     <1> rename_df_compare_sf_df_name:
5105 000095F4 BE[44800100]               <1>     mov     esi, FindFile_Name
5106 000095F9 BF[8E810100]               <1>     mov     edi, SourceFile_Name
5107                                     <1>     ;mov     ecx, 12
5108                                     <1>     ; 28/07/2022
5109 000095FE 29C9                       <1>     sub     ecx, ecx
5110 00009600 B10C                       <1>     mov     cl, 12
5111                                     <1> rename_df_compare_sf_df_name_next:
5112 00009602 AC                         <1>     lodsb
5113 00009603 AE                         <1>     scasb
5114 00009604 7506                       <1>     jne     short loc_rename_df_find
5115 00009606 08C0                       <1>     or      al, al
5116 00009608 74A9                       <1>     jz      short loc_rename_df_cmd_failed
5117 0000960A E2F6                       <1>     loop    rename_df_compare_sf_df_name_next
5118                                     <1>
5119                                     <1> loc_rename_df_find:
5120                                     <1>     ;mov     esi, [DelFile_FNPointer]
5121 0000960C BE[44800100]               <1>     mov     esi, FindFile_Name
5122                                     <1>
5123                                     <1>     ;xor     ax, ax ; Any
5124                                     <1>     ; 28/07/2022
5125 00009611 31C0                       <1>     xor     eax, eax ; 0 ; Any
5126 00009613 E852F3FFFF                 <1>     call    find_first_file
5127                                     <1>     ;jnc     short loc_rename_df_found
5128                                     <1>     ; 29/12/2017
5129                                     <1>     ;jnc     loc_permission_denied
5130                                     <1>     ; 28/07/2022
5131 00009618 7205                       <1>     jc      short loc_rename_df_check_error_code
5132 0000961A E923F7FFFF                 <1>     jmp     loc_permission_denied
5133                                     <1>
5134                                     <1> loc_rename_df_check_error_code:
5135                                     <1>     ;cmp     eax, 2
5136 0000961F 3C02                       <1>     cmp     al, 2 ; Not found error
5137 00009621 7406                       <1>     je      short rename_df_move_find_struct_to_dest
5138 00009623 F9                         <1>     stc
5139 00009624 E90FF7FFFF                 <1>     jmp     loc_file_rw_cmd_failed
5140                                     <1>
5141                                     <1> ;loc_rename_df_found:
5142                                     <1>     ; 05/11/2016
5143                                     <1>     ; Permission denied error
5144                                     <1>     ;mov     eax, ERR_PERM_DENIED ; 29/12/2017
5145                                     <1>     ;stc
5146                                     <1>     ;jmp     loc_permission_denied ; 06/11/2016
5147                                     <1>
5148                                     <1> rename_df_move_find_struct_to_dest:
5149 00009629 BE[02800100]               <1>     mov     esi, FindFile_Drv
5150 0000962E BF[CC810100]               <1>     mov     edi, DestinationFile_Drv
5151                                     <1>     ;mov     ecx, 32
5152                                     <1>     ; 28/07/2022
5153 00009633 29C9                       <1>     sub     ecx, ecx
5154 00009635 B120                       <1>     mov     cl, 32
5155 00009637 F3A5                       <1>     rep     movsd
5156                                     <1>
5157                                     <1> loc_rename_df_process_q_sf:
5158                                     <1>     ;mov     ecx, 12
5159 00009639 B10C                       <1>     mov     cl, 12
5160 0000963B BE[8E810100]               <1>     mov     esi, SourceFile_Name
5161 00009640 BF[D6330100]               <1>     mov     edi, Rename_OldName
5162                                     <1> rename_df_process_q_nml_1_sf:
5163 00009645 AC                         <1>     lodsb
5164 00009646 3C20                       <1>     cmp     al, 20h
5165 00009648 7603                       <1>     jna     short rename_df_process_q_nml_2_sf
5166 0000964A AA                         <1>     stosb
5167 0000964B E2F8                       <1>     loop    rename_df_process_q_nml_1_sf
5168                                     <1>
5169                                     <1> rename_df_process_q_nml_2_sf:
5170 0000964D C60700                     <1>     mov     byte [edi], 0
5171                                     <1>
5172                                     <1> loc_rename_df_process_q_df:
5173                                     <1>     ;mov     ecx, 12
5174 00009650 B10C                       <1>     mov     cl, 12
5175 00009652 BE[0E820100]               <1>     mov     esi, DestinationFile_Name
5176 00009657 BF[E7330100]               <1>     mov     edi, Rename_NewName
5177                                     <1> rename_df_process_q_nml_1_df:
5178 0000965C AC                         <1>     lodsb
5179 0000965D 3C20                       <1>     cmp     al, 20h
5180 0000965F 7603                       <1>     jna     short loc_rename_df_process_q_nml_2_df
5181 00009661 AA                         <1>     stosb
5182 00009662 E2F8                       <1>     loop    rename_df_process_q_nml_1_df
5183                                     <1>
5184                                     <1> loc_rename_df_process_q_nml_2_df:
5185 00009664 C60700                     <1>     mov     byte [edi], 0
5186                                     <1>

```

```

5187                                     <1> loc_rename_confirmation_question:
5188 00009667 BE[AE330100]             <1>     mov     esi, Msg_DoYouWantRename
5189 0000966C E839D7FFFF             <1>     call    print_msg
5190                                     <1>
5191 00009671 A0[A9810100]             <1>     mov     al, [SourceFile_DirEntry+11] ; Attributes
5192 00009676 2410                   <1>     and     al, 10h
5193 00009678 750C                   <1>     jnz     short rename_confirmation_question_dir
5194                                     <1>
5195                                     <1> rename_confirmation_question_file:
5196 0000967A BE[C5330100]             <1>     mov     esi, Rename_File
5197 0000967F E826D7FFFF             <1>     call    print_msg
5198 00009684 EB0A                   <1>     jmp     short rename_confirmation_question_as
5199                                     <1>
5200                                     <1> rename_confirmation_question_dir:
5201 00009686 BE[CB330100]             <1>     mov     esi, Rename_Directory
5202 0000968B E81AD7FFFF             <1>     call    print_msg
5203                                     <1>
5204                                     <1> rename_confirmation_question_as:
5205 00009690 BE[D6330100]             <1>     mov     esi, Rename_OldName
5206 00009695 E810D7FFFF             <1>     call    print_msg
5207 0000969A BE[E3330100]             <1>     mov     esi, Msg_File_rename_as
5208 0000969F E806D7FFFF             <1>     call    print_msg
5209 000096A4 BE[0A330100]             <1>     mov     esi, Msg_YesNo
5210 000096A9 E8FCD6FFFF             <1>     call    print_msg
5211                                     <1>
5212                                     <1> loc_rename_ask_again:
5213 000096AE 30E4                   <1>     xor     ah, ah
5214 000096B0 E86778FFFF             <1>     call    int16h
5215 000096B5 3C1B                   <1>     cmp     al, 1Bh
5216 000096B7 740F                   <1>     je      short loc_do_not_rename_file
5217 000096B9 24DF                   <1>     and     al, 0DFh
5218 000096BB A2[14330100]             <1>     mov     [Y_N_nextline], al
5219 000096C0 3C59                   <1>     cmp     al, 'y'
5220 000096C2 7404                   <1>     je      short loc_yes_rename_file
5221 000096C4 3C4E                   <1>     cmp     al, 'N'
5222 000096C6 75E6                   <1>     jne     short loc_rename_ask_again
5223                                     <1>
5224                                     <1> loc_do_not_rename_file:
5225                                     <1> loc_yes_rename_file:
5226 000096C8 E89EF7FFFF             <1>     call    y_n_answer ; 29/12/2017
5227                                     <1>     ;cmp     al, 'y' ; 'yes'
5228                                     <1>     ;cmc
5229                                     <1>     ;jnc     loc_file_rw_restore_retn
5230 000096CD 3C4E                   <1>     cmp     al, 'N' ; 'no'
5231                                     <1>     ;je      loc_file_rw_restore_retn
5232                                     <1>     ; 28/07/2022
5233 000096CF 7505                   <1>     jne     short loc_rename_file_yes
5234 000096D1 E962F6FFFF             <1>     jmp     loc_file_rw_restore_retn
5235                                     <1>
5236                                     <1> loc_rename_file_yes: ; 28/07/2022
5237 000096D6 BE[E7330100]             <1>     mov     esi, Rename_NewName
5238 000096DB 668B0D[C6810100]         <1>     mov     cx, [SourceFile_DirEntryNumber]
5239 000096E2 66A1[B2810100]             <1>     mov     ax, [SourceFile_DirEntry+20] ; First Cluster, HW
5240 000096E8 C1E010                   <1>     shl     eax, 16 ; 13/11/2017
5241 000096EB 66A1[B8810100]             <1>     mov     ax, [SourceFile_DirEntry+26] ; First Cluster, LW
5242                                     <1>
5243 000096F1 0FB61D[9B810100]         <1>     movzx   ebx, byte [SourceFile_LongNameEntryLength]
5244 000096F8 E8201A0000             <1>     call    rename_directory_entry
5245 000096FD E94FF7FFFF             <1>     jmp     loc_rename_file_ok
5246                                     <1> ;loc_rename_file_ok:
5247                                     <1> ;     jc      loc_run_cmd_failed
5248                                     <1> ;     mov     esi, Msg_OK
5249                                     <1> ;     call    proc_printmsg
5250                                     <1> ;     jmp     loc_file_rw_restore_retn
5251                                     <1>
5252                                     <1> move_file:
5253                                     <1> ; 07/08/2022
5254                                     <1> ; 28/07/2022 (TRDOS 386 Kernel v2.0.5)
5255                                     <1> ; 11/03/2016
5256                                     <1> ; 09/03/2016
5257                                     <1> ; 08/03/2016 (TRDOS 386 = TRDOS v2.0)
5258                                     <1> ; 21/05/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_move')
5259                                     <1> ; 23/04/2011
5260                                     <1>
5261                                     <1> get_move_source_fchar:
5262                                     <1> ; esi = file name
5263 00009702 803E20             <1>     cmp     byte [esi], 20h
5264 00009705 7613                   <1>     jna     short loc_move_nofilename_retn
5265                                     <1>
5266 00009707 8935[44810100]         <1>     mov     [SourceFilePath], esi
5267                                     <1>
5268                                     <1> move_scan_source_file:
5269 0000970D 46                   <1>     inc     esi
5270 0000970E 803E20             <1>     cmp     byte [esi], 20h
5271 00009711 7408                   <1>     je      short move_scan_destination_1
5272                                     <1>     ;jb     short loc_move_nofilename_retn
5273                                     <1>     ;jb     loc_cmd_failed
5274                                     <1>     ;jmp     short move_scan_source_file
5275                                     <1>     ; 28/07/2022
5276 00009713 77F8                   <1>     ja      short move_scan_source_file
5277                                     <1> loc_move_failed:
5278 00009715 E92AEDFFFF             <1>     jmp     loc_cmd_failed
5279                                     <1>
5280                                     <1> loc_move_nofilename_retn:
5281 0000971A C3                   <1>     retn
5282                                     <1>
5283                                     <1> move_scan_destination_1:
5284 0000971B C60600             <1>     mov     byte [esi], 0
5285                                     <1>
5286                                     <1> move_scan_destination_2:
5287 0000971E 46                   <1>     inc     esi
5288 0000971F 803E20             <1>     cmp     byte [esi], 20h
5289 00009722 74FA                   <1>     je      short move_scan_destination_2
5290                                     <1>     ;jb     short loc_move_nofilename_retn
5291                                     <1>     ;jb     loc_cmd_failed
5292                                     <1>     ; 28/07/2022
5293 00009724 72EF                   <1>     jb      short loc_move_failed
5294                                     <1>
5295 00009726 8935[48810100]         <1>     mov     [DestinationFilePath], esi
5296                                     <1>
5297                                     <1> move_scan_destination_3:
5298 0000972C 46                   <1>     inc     esi
5299 0000972D 803E20             <1>     cmp     byte [esi], 20h
5300 00009730 77FA                   <1>     ja      short move_scan_destination_3
5301 00009732 C60600             <1>     mov     byte [esi], 0
5302                                     <1>
5303                                     <1> loc_move_scan_destination_OK:
5304 00009735 8B35[44810100]         <1>     mov     esi, [SourceFilePath]
5305 0000973B 8B3D[48810100]         <1>     mov     edi, [DestinationFilePath]
5306                                     <1>
5307 00009741 B001                   <1>     mov     al, 1 ; move procedure Phase 1
5308 00009743 E8411A0000             <1>     call    move_source_file_to_destination_file
5309 00009748 7325                   <1>     jnc     short move_source_file_to_destination_question
5310                                     <1>

```

```

5311 <1> loc_move_cmd_failed_1:
5312 0000974A 08C0 <1> or al, al
5313 <1> ;jz loc_cmd_failed
5314 <1> ; 28/07/2022
5315 0000974C 74C7 <1> jz short loc_move_failed
5316 <1>
5317 0000974E 3C11 <1> cmp al, 11h
5318 00009750 7409 <1> je short loc_msg_not_same_device
5319 <1> ;cmp al, 05h
5320 <1> ;cmp al, ERR_PERM_DENIED ; 29/12/2017
5321 <1> ;jne loc_run_cmd_failed
5322 <1> ;jmp loc_permission_denied
5323 00009752 3C0B <1> cmp al, ERR_PERM_DENIED
5324 <1> ;je loc_permission_denied
5325 <1> ; 28/07/2022
5326 00009754 7414 <1> je short loc_move_perm_denied
5327 00009756 E914EDFFFF <1> jmp loc_run_cmd_failed
5328 <1>
5329 <1> ;mov esi, Msg_Permission_denied
5330 <1> ;call print_msg
5331 <1> ;jmp loc_file_rw_restore_retn
5332 <1>
5333 <1> loc_msg_not_same_device:
5334 0000975B BE[F4330100] <1> mov esi, msg_not_same_drv
5335 00009760 E845D6FFFF <1> call print_msg
5336 00009765 E9CEF5FFFF <1> jmp loc_file_rw_restore_retn
5337 <1>
5338 <1> ; 28/07/2022
5339 <1> loc_move_perm_denied:
5340 0000976A E9D3F5FFFF <1> jmp loc_permission_denied
5341 <1>
5342 <1> move_source_file_to_destination_question:
5343 0000976F A0[4C810100] <1> mov al, [SourceFile_Drv]
5344 00009774 0441 <1> add al, 'A'
5345 00009776 A2[56340100] <1> mov [msg_source_file_drv], al
5346 0000977B A0[CC810100] <1> mov al, [DestinationFile_Drv]
5347 00009780 0441 <1> add al, 'A'
5348 00009782 A2[75340100] <1> mov [msg_destination_file_drv], al
5349 <1>
5350 00009787 57 <1> push edi ; *
5351 <1>
5352 00009788 BE[3A340100] <1> mov esi, msg_source_file
5353 0000978D E818D6FFFF <1> call print_msg
5354 00009792 BE[4D810100] <1> mov esi, SourceFile_Directory
5355 00009797 803E20 <1> cmp byte [esi], 20h
5356 0000979A 7605 <1> jna short msftdfq_sfn
5357 0000979C E809D6FFFF <1> call print_msg
5358 <1> msftdfq_sfn:
5359 000097A1 BE[8E810100] <1> mov esi, SourceFile_Name
5360 000097A6 E8FFD5FFFF <1> call print_msg
5361 000097AB BE[59340100] <1> mov esi, msg_destination_file
5362 000097B0 E8F5D5FFFF <1> call print_msg
5363 000097B5 BE[CD810100] <1> mov esi, DestinationFile_Directory
5364 000097BA 803E20 <1> cmp byte [esi], 20h
5365 000097BD 7605 <1> jna short msftdfq_dfn
5366 000097BF E8E6D5FFFF <1> call print_msg
5367 <1> msftdfq_dfn:
5368 000097C4 BE[0E820100] <1> mov esi, DestinationFile_Name
5369 000097C9 E8DCD5FFFF <1> call print_msg
5370 000097CE BE[78340100] <1> mov esi, msg_copy_nextline
5371 000097D3 E8D2D5FFFF <1> call print_msg
5372 000097D8 BE[78340100] <1> mov esi, msg_copy_nextline
5373 000097DD E8C8D5FFFF <1> call print_msg
5374 <1>
5375 <1> loc_move_ask_for_new_file_yes_no:
5376 000097E2 BE[06340100] <1> mov esi, Msg_DoYouWantMoveFile
5377 000097E7 E8BED5FFFF <1> call print_msg
5378 000097EC BE[0A330100] <1> mov esi, Msg_YesNo
5379 000097F1 E8B4D5FFFF <1> call print_msg
5380 <1> loc_move_ask_for_new_file_again:
5381 000097F6 30E4 <1> xor ah, ah
5382 000097F8 E81F77FFFF <1> call int16h
5383 000097FD 3C1B <1> cmp al, 1Bh
5384 <1> ;je short loc_do_not_move_file
5385 000097FF 743F <1> je short loc_move_y_n_escape
5386 00009801 24DF <1> and al, 0DFh
5387 00009803 A2[14330100] <1> mov [Y_N_nextline], al
5388 00009808 3C59 <1> cmp al, 'Y'
5389 0000980A 7404 <1> je short loc_yes_move_file
5390 0000980C 3C4E <1> cmp al, 'N'
5391 0000980E 75E6 <1> jne short loc_move_ask_for_new_file_again
5392 <1>
5393 <1> loc_do_not_move_file:
5394 <1> loc_yes_move_file:
5395 00009810 E856F6FFFF <1> call y_n_answer ; 29/12/2017
5396 00009815 5F <1> pop edi ; *
5397 <1> ;cmp al, 'Y' ; 'yes'
5398 <1> ;cmc
5399 <1> ;jnc loc_file_rw_restore_retn
5400 00009816 3C4E <1> cmp al, 'N' ; 'no'
5401 <1> ;je loc_file_rw_restore_retn
5402 <1> ; 28/07/2022
5403 00009818 7421 <1> je short loc_move_rw_restore_retn
5404 <1>
5405 <1> loc_move_yes_move_file:
5406 0000981A B002 <1> mov al, 2 ; move procedure Phase 2
5407 0000981C E868190000 <1> call move_source_file_to_destination_file
5408 <1> ;jc short loc_move_cmd_failed_2
5409 <1> ;jnc move_source_file_to_dest_OK
5410 <1> ; 28/07/2022
5411 00009821 7205 <1> jc short loc_move_cmd_failed_2
5412 <1> ; 07/08/2022
5413 00009823 E930F6FFFF <1> jmp move_source_file_to_dest_OK
5414 <1>
5415 <1> ;move_source_file_to_destination_OK:
5416 <1> ; mov esi, Msg_OK
5417 <1> ; call print_msg
5418 <1> ; jmp loc_file_rw_restore_retn
5419 <1>
5420 <1> loc_move_cmd_failed_2:
5421 00009828 3C27 <1> cmp al, 27h
5422 <1> ;jne loc_run_cmd_failed
5423 <1> ; 28/07/2022
5424 0000982A 7405 <1> je short loc_move_ids_err
5425 0000982C E93EECFFFF <1> jmp loc_run_cmd_failed
5426 <1>
5427 <1> loc_move_ids_err: ; 28/07/2022
5428 00009831 BE[1F340100] <1> mov esi, msg_insufficient_disk_space
5429 00009836 E86FD5FFFF <1> call print_msg
5430 <1>
5431 <1> loc_move_rw_restore_retn: ; 28/07/2022
5432 0000983B E9F8F4FFFF <1> jmp loc_file_rw_restore_retn
5433 <1>
5434 <1> loc_move_y_n_escape:

```

```

5435 00009840 B04E      <1>      mov     al, 'N' ; 'no'
5436 00009842 EBCC      <1>      jmp     short loc_do_not_move_file
5437                    <1>
5438                    <1> copy_file:
5439                    <1>      ; 31/08/2024 - TRDOS 386 v2.0.9
5440                    <1>      ; 25/07/2022 - TRDOS 386 kernel v2.0.5
5441                    <1>      ; 15/10/2016
5442                    <1>      ; 24/03/2016
5443                    <1>      ; 21/03/2016
5444                    <1>      ; 15/03/2016 (TRDOS 386 = TRDOS v2.0)
5445                    <1>      ; 21/05/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_copy')
5446                    <1>      ; 01/08/2010
5447                    <1>
5448                    <1> get_copy_source_fchar:
5449                    <1>      ; esi = file name
5450 00009844 803E20      <1>      cmp     byte [esi], 20h
5451 00009847 7613      <1>      jna     short loc_copy_nofilename_retn
5452                    <1>
5453 00009849 8935[44810100] <1>      mov     [SourceFilePath], esi
5454                    <1>
5455                    <1> copy_scan_source_file:
5456 0000984F 46          <1>      inc     esi
5457 00009850 803E20      <1>      cmp     byte [esi], 20h
5458 00009853 7408      <1>      je      short copy_scan_destination_1
5459                    <1>      ;;;jb short loc_copy_nofilename_retn
5460                    <1>      ;jb  loc_cmd_failed
5461                    <1>      ;jmp short copy_scan_source_file
5462                    <1>      ; 25/07/2022
5463 00009855 73F8      <1>      jnb     short copy_scan_source_file
5464                    <1> copy_scan_destination_0:
5465 00009857 E9E8EBFFFF      <1>      jmp     loc_cmd_failed
5466                    <1>
5467                    <1> loc_copy_nofilename_retn:
5468 0000985C C3          <1>      retn
5469                    <1>
5470                    <1> copy_scan_destination_1:
5471 0000985D C60600      <1>      mov     byte [esi], 0
5472                    <1>
5473                    <1> copy_scan_destination_2:
5474 00009860 46          <1>      inc     esi
5475 00009861 803E20      <1>      cmp     byte [esi], 20h
5476 00009864 74FA      <1>      je      short copy_scan_destination_2
5477                    <1>      ;;;jb short loc_copy_nofilename_retn
5478                    <1>      ;jb  loc_cmd_failed
5479                    <1>      ; 25/07/2022
5480 00009866 72EF      <1>      jnb     short copy_scan_destination_0
5481                    <1>
5482 00009868 8935[48810100] <1>      mov     [DestinationFilePath], esi
5483                    <1>
5484                    <1> copy_scan_destination_3:
5485 0000986E 46          <1>      inc     esi
5486 0000986F 803E20      <1>      cmp     byte [esi], 20h
5487 00009872 77FA      <1>      ja      short copy_scan_destination_3
5488 00009874 C60600      <1>      mov     byte [esi], 0
5489                    <1>
5490                    <1> loc_copy_save_current_drive:
5491 00009877 8A35[62770100] <1>      mov     dh, [Current_Drv]
5492 0000987D 8835[B7E0100] <1>      mov     [RUN_CDRV], dh
5493                    <1>
5494                    <1> copy_source_file_to_destination_phase_1:
5495 00009883 8B35[44810100] <1>      mov     esi, [SourceFilePath]
5496 00009889 8B3D[48810100] <1>      mov     edi, [DestinationFilePath]
5497                    <1>
5498 0000988F B001      <1>      mov     al, 1 ; copy procedure Phase 1
5499 00009891 E8571B0000      <1>      call    copy_source_file_to_destination_file
5500 00009896 7327      <1>      jnc     short copy_source_file_to_destination_question
5501                    <1>
5502                    <1> loc_copy_cmd_failed_1:
5503                    <1>      ; 18/03/2016 (restore current drive and directory)
5504 00009898 08C0      <1>      or      al, al
5505 0000989A 7507      <1>      jnz     short loc_copy_cmd_failed_2
5506                    <1>
5507 0000989C FEC0      <1>      inc     al ; mov al, 1 ; Bad command or file name !
5508                    <1>      ; 25/07/2022
5509 0000989E E9CCEBFFFF      <1>      jmp     loc_run_cmd_failed
5510                    <1>
5511                    <1> loc_copy_cmd_failed_2:
5512 000098A3 3C27      <1>      cmp     al, 27h ; Insufficient disk space
5513 000098A5 7409      <1>      je      short loc_file_write_insuff_disk_space_msg
5514                    <1>
5515                    <1>      ; 29/12/2017
5516                    <1>      ;cmp al, 05h
5517 000098A7 3C0B      <1>      cmp     al, ERR_PERM_DENIED
5518                    <1>      ;jne loc_run_cmd_failed
5519                    <1>      ; 25/07/2022
5520 000098A9 75F3      <1>      jne     short loc_copy_cmd_failed_3
5521                    <1>
5522 000098AB E992F4FFFF      <1>      jmp     loc_permission_denied
5523                    <1>
5524                    <1> loc_file_write_insuff_disk_space_msg:
5525 000098B0 BE[1F340100] <1>      mov     esi, msg_insufficient_disk_space
5526 000098B5 E8F0D4FFFF      <1>      call    print_msg
5527 000098BA E979F4FFFF      <1>      jmp     loc_file_rw_restore_retn
5528                    <1>
5529                    <1> copy_source_file_to_destination_question:
5530 000098BF 57          <1>      push    edi ; *
5531                    <1>
5532                    <1>      ; dh = source file attributes
5533                    <1>      ; dl > 0 -> destination file found
5534 000098C0 20D2      <1>      and     dl, dl
5535 000098C2 7446      <1>      jz      short copy_source_file_to_destination_pass_owrq
5536                    <1>
5537                    <1> loc_copy_ask_for_owr_yes_no:
5538 000098C4 BE[7B340100] <1>      mov     esi, Msg_DoYouWantOverwriteFile
5539 000098C9 E8DCD4FFFF      <1>      call    print_msg
5540 000098CE BE[0E820100] <1>      mov     esi, DestinationFile_Name
5541 000098D3 E8D2D4FFFF      <1>      call    print_msg
5542 000098D8 BE[0A330100] <1>      mov     esi, Msg_YesNo
5543 000098DD E8C8D4FFFF      <1>      call    print_msg
5544                    <1>
5545                    <1> loc_copy_ask_for_owr_again:
5546 000098E2 30E4      <1>      xor     ah, ah
5547 000098E4 E83376FFFF      <1>      call    int16h
5548 000098E9 3C1B      <1>      cmp     al, 1Bh
5549                    <1>      ;je loc_do_not_copy_file
5550 000098EB 7419      <1>      je      short loc_copy_y_n_escape
5551 000098ED 24DF      <1>      and     al, 0DFh
5552 000098EF A2[14330100] <1>      mov     [Y_N_nextline], al
5553 000098F4 3C59      <1>      cmp     al, 'Y'
5554                    <1>      ;je loc_yes_copy_file
5555                    <1>      ; 25/07/2022
5556 000098F6 7505      <1>      jne     short loc_copy_ask_for_owr_n
5557 000098F8 E9AD000000      <1>      jmp     loc_yes_copy_file
5558                    <1>

```

```

5559 <1> loc_copy_ask_for_owr_n: ; 25/07/2022
5560 000098FD 3C4E <1> cmp al, 'N'
5561 <1> ;je loc_do_not_copy_file
5562 <1> ;jmp short loc_copy_ask_for_owr_again
5563 <1> ; 25/07/2022
5564 000098FF 75E1 <1> jne short loc_copy_ask_for_owr_again
5565 <1> loc_do_not_copy_file_j:
5566 00009901 E9A4000000 <1> jmp loc_do_not_copy_file
5567 <1>
5568 <1> loc_copy_y_n_escape:
5569 00009906 B04E <1> mov al, 'N' ; 'no'
5570 <1> ;jmp loc_do_not_copy_file
5571 <1> ; 25/07/2022
5572 00009908 EBF7 <1> jmp short loc_do_not_copy_file_j
5573 <1>
5574 <1> copy_source_file_to_destination_pass_owrq:
5575 0000990A A0[4C810100] <1> mov al, [SourceFile_Drv]
5576 0000990F 0441 <1> add al, 'A'
5577 00009911 A2[56340100] <1> mov [msg_source_file_drv], al
5578 00009916 A0[CC810100] <1> mov al, [DestinationFile_Drv]
5579 0000991B 0441 <1> add al, 'A'
5580 0000991D A2[75340100] <1> mov [msg_destination_file_drv], al
5581 <1>
5582 00009922 BE[3A340100] <1> mov esi, msg_source_file
5583 00009927 E87ED4FFFF <1> call print_msg
5584 0000992C BE[4D810100] <1> mov esi, SourceFile_Directory
5585 00009931 803E20 <1> cmp byte [esi], 20h
5586 00009934 7605 <1> jna short csftdfq_sfn
5587 00009936 E86FD4FFFF <1> call print_msg
5588 <1> csftdfq_sfn:
5589 0000993B BE[8E810100] <1> mov esi, SourceFile_Name
5590 00009940 E865D4FFFF <1> call print_msg
5591 00009945 BE[59340100] <1> mov esi, msg_destination_file
5592 0000994A E85BD4FFFF <1> call print_msg
5593 0000994F BE[CD810100] <1> mov esi, DestinationFile_Directory
5594 00009954 803E20 <1> cmp byte [esi], 20h
5595 00009957 7605 <1> jna short csftdfq_dfn
5596 00009959 E84CD4FFFF <1> call print_msg
5597 <1> csftdfq_dfn:
5598 0000995E BE[0E820100] <1> mov esi, DestinationFile_Name
5599 00009963 E842D4FFFF <1> call print_msg
5600 00009968 BE[78340100] <1> mov esi, msg_copy_nextline
5601 0000996D E838D4FFFF <1> call print_msg
5602 00009972 BE[78340100] <1> mov esi, msg_copy_nextline
5603 00009977 E82ED4FFFF <1> call print_msg
5604 <1>
5605 <1> loc_copy_ask_for_new_file_yes_no:
5606 0000997C BE[9A340100] <1> mov esi, Msg_DoYouWantCopyFile
5607 00009981 E824D4FFFF <1> call print_msg
5608 00009986 BE[0A330100] <1> mov esi, Msg_YesNo
5609 0000998B E81AD4FFFF <1> call print_msg
5610 <1>
5611 <1> loc_copy_ask_for_new_file_again:
5612 00009990 30E4 <1> xor ah, ah
5613 00009992 E88575FFFF <1> call int16h
5614 00009997 3C1B <1> cmp al, 1Bh
5615 00009999 740F <1> je short loc_do_not_copy_file
5616 0000999B 24DF <1> and al, 0DFh
5617 0000999D A2[14330100] <1> mov [Y_N_nextline], al
5618 000099A2 3C59 <1> cmp al, 'Y'
5619 000099A4 7404 <1> je short loc_yes_copy_file
5620 000099A6 3C4E <1> cmp al, 'N'
5621 000099A8 75E6 <1> jne short loc_copy_ask_for_new_file_again
5622 <1>
5623 <1> loc_do_not_copy_file:
5624 <1> loc_yes_copy_file:
5625 000099AA E8BCF4FFFF <1> call y_n_answer ; 29/12/2017
5626 000099AF 5F <1> pop edi ; *
5627 <1> ;cmp al, 'y' ; 'yes'
5628 <1> ;cmc
5629 <1> ;jnc loc_file_rw_restore_retn
5630 000099B0 3C4E <1> cmp al, 'N' ; 'no'
5631 <1> ;je loc_file_rw_restore_retn
5632 <1> ; 25/07/2022
5633 000099B2 7505 <1> jne short copy_source_file_to_destination_pass_q
5634 000099B4 E97FF3FFFF <1> jmp loc_file_rw_restore_retn
5635 <1>
5636 <1> copy_source_file_to_destination_pass_q:
5637 000099B9 B002 <1> mov al, 2 ; copy procedure Phase 2
5638 000099BB E82D1A0000 <1> call copy_source_file_to_destination_file
5639 <1> ;jc short loc_file_write_check_disk_space_err
5640 <1>
5641 <1> ; 31/08/2024
5642 000099C0 9C <1> pushf
5643 <1>
5644 <1> ; 24/03/2016
5645 <1> ;;push cx
5646 <1> ;push ecx ; 29/12/2017
5647 <1>
5648 000099C1 BE[78340100] <1> mov esi, msg_copy_nextline
5649 000099C6 E8DFD3FFFF <1> call print_msg
5650 <1>
5651 <1> ;pop eax ; 29/12/2017
5652 <1> ;;pop cx
5653 <1> ;pop ax
5654 <1>
5655 <1> ; 31/08/2024
5656 000099CB 9D <1> popf
5657 000099CC 7305 <1> jnc short copy_source_file_to_destination_OK
5658 <1>
5659 <1> ; 31/08/2024
5660 <1> ;;or cl, cl
5661 <1> ;;or al, al
5662 <1> ;jz short copy_source_file_to_destination_OK
5663 <1> ;
5664 <1> ;; 15/10/2016 (1Dh -> 18)
5665 <1> ;; 18/03/2016 (1Dh)
5666 <1> ;;cmp cl, 18 ; write error
5667 <1> ;cmp al, 18
5668 <1> ;jne short copy_source_file_to_destination_not_OK
5669 <1> ;
5670 <1> ;mov al, cl ; error number (write fault!)
5671 <1> ;stc
5672 <1>
5673 000099CE E965F3FFFF <1> jmp loc_file_rw_cmd_failed
5674 <1>
5675 <1> ; 31/08/2024
5676 <1> ;copy_source_file_to_destination_not_OK:
5677 <1> ;mov esi, Msg_read_file_error_before_EOF
5678 <1> ;call print_msg
5679 <1> ;jmp loc_file_rw_restore_retn
5680 <1>
5681 <1> copy_source_file_to_destination_OK:
5682 000099D3 BE[18330100] <1> mov esi, Msg_OK

```



```

5683 000099D8 E8CDD3FFFF <1> call print_msg
5684 <1>
5685 000099DD E956F3FFFF <1> jmp loc_file_rw_restore_retn
5686 <1>
5687 <1> ;loc_file_write_check_disk_space_err:
5688 <1> ;cmp al, 27h ; Insufficient disk space
5689 <1> ;je loc_file_write_insuff_disk_space_msg
5690 <1> ;jb loc_file_rw_cmd_failed
5691 <1>
5692 <1> ;call print_misc_error_msg ; 15/03/2016
5693 <1> ;jmp loc_file_rw_restore_retn
5694 <1>
5695 <1> ; 19/12/2025
5696 <1> %if 0
5697 <1> change_fs_file_attributes:
5698 <1> ; 04/03/2016 ; Temporary
5699 <1> ; AL = File or directory attributes
5700 <1> ; AH = 0 -> Attributes are in MS-DOS format
5701 <1> ; AH > 0 -> Attributes are in SINGLIX format
5702 <1> ;push ebx
5703 <1> ; ... do somethings here ...
5704 <1> ;pop ebx
5705 <1> ; BL = File or directory attributes
5706 <1> retn
5707 <1> %endif
5708 <1>
5709 <1> set_get_env:
5710 <1> ; 25/07/2022 - TRDOS 386 Kernel v2.0.5
5711 <1> ; 11/04/2016 (TRDOS 386 = TRDOS v2.0)
5712 <1> ; 02/09/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_set')
5713 <1> ; 2005 - 28/08/2011
5714 <1> get_setenv_fchar:
5715 <1> ; esi = environment variable/string
5716 000099E2 8A06 <1> mov al, [esi]
5717 000099E4 3C20 <1> cmp al, 20h
5718 000099E6 771E <1> ja short loc_find_env
5719 <1>
5720 000099E8 BE00300900 <1> mov esi, Env_Page
5721 <1> loc_print_setline:
5722 000099ED 803E00 <1> cmp byte [esi], 0
5723 000099F0 7613 <1> jna short loc_setenv_retn
5724 000099F2 E8B3D3FFFF <1> call print_msg
5725 000099F7 56 <1> push esi
5726 000099F8 BE[07390100] <1> mov esi, nextline
5727 000099FD E8A8D3FFFF <1> call print_msg
5728 00009A02 5E <1> pop esi
5729 00009A03 EBE8 <1> jmp short loc_print_setline
5730 <1>
5731 <1> loc_setenv_retn:
5732 00009A05 C3 <1> retn
5733 <1>
5734 <1> loc_find_env:
5735 00009A06 3C3D <1> cmp al, '='
5736 <1> ;je loc_cmd_failed
5737 <1> ; 25/07/2022
5738 00009A08 7505 <1> jne short loc_find_envr
5739 00009A0A E935EAFFFF <1> jmp loc_cmd_failed
5740 <1>
5741 <1> loc_find_envr: ; 25/07/2022
5742 00009A0F 56 <1> push esi
5743 <1> loc_repeat_env_equal_check:
5744 00009A10 46 <1> inc esi
5745 00009A11 803E3D <1> cmp byte [esi], '='
5746 00009A14 7430 <1> je short pass_env_equal_check
5747 00009A16 803E20 <1> cmp byte [esi], 20h
5748 00009A19 73F5 <1> jnb short loc_repeat_env_equal_check
5749 00009A1B C60600 <1> mov byte [esi], 0
5750 00009A1E 5E <1> pop esi
5751 <1> ; 25/07/2022 (*)
5752 <1> loc_print_env_string:
5753 00009A1F BF[62780100] <1> mov edi, TextBuffer ; out buffer
5754 00009A24 B9FF000000 <1> mov ecx, 255 ; maximum size (limit)
5755 00009A29 30C0 <1> xor al, al ; 0 -> use [ESI]
5756 00009A2B E877000000 <1> call get_environment_string
5757 00009A30 72D3 <1> jc short loc_setenv_retn
5758 <1> ; 25/07/2022
5759 <1> ;loc_print_env_string:
5760 00009A32 BE[62780100] <1> mov esi, TextBuffer
5761 00009A37 E86ED3FFFF <1> call print_msg
5762 00009A3C BE[07390100] <1> mov esi, nextline
5763 <1> ;call print_msg
5764 <1> ;retn
5765 <1> ; 25/07/2022
5766 00009A41 E964D3FFFF <1> jmp print_msg
5767 <1>
5768 <1> pass_env_equal_check:
5769 00009A46 46 <1> inc esi
5770 00009A47 803E20 <1> cmp byte [esi], 20h
5771 00009A4A 73FA <1> jnb short pass_env_equal_check
5772 00009A4C C60600 <1> mov byte [esi], 0
5773 <1>
5774 <1> loc_call_set_env_string:
5775 00009A4F 5E <1> pop esi
5776 00009A50 E815010000 <1> call set_environment_string
5777 00009A55 73AE <1> jnc short loc_setenv_retn
5778 <1>
5779 <1> loc_set_cmd_failed:
5780 00009A57 3C08 <1> cmp al, 08h
5781 <1> ;jne loc_cmd_failed
5782 <1> ; 25/07/2022
5783 00009A59 7405 <1> je short loc_set_cmd_failed_spc
5784 00009A5B E9E4E9FFFF <1> jmp loc_cmd_failed
5785 <1>
5786 <1> loc_set_cmd_failed_spc:
5787 00009A60 BE[F3340100] <1> mov esi, Msg_No_Set_Space
5788 <1> ;call print_msg
5789 <1> ;retn
5790 <1> ; 25/07/2022
5791 00009A65 E940D3FFFF <1> jmp print_msg
5792 <1>
5793 <1> set_get_path:
5794 <1> ; 25/07/2022 - TRDOS 386 Kernel v2.0.5
5795 <1> ; 11/04/2016 (TRDOS 386 = TRDOS v2.0)
5796 <1> ; 03/09/2011 (TRDOS v1, CMD_INTR.ASM, 'cmp_cmd_path')
5797 <1> ; 2005
5798 <1> get_path_fchar:
5799 <1> ; esi = path
5800 00009A6A 803E20 <1> cmp byte [esi], 20h
5801 00009A6D 7711 <1> ja short loc_set_path
5802 <1>
5803 00009A6F BE00300900 <1> mov esi, Env_Page
5804 <1> loc_print_path:
5805 00009A74 803E00 <1> cmp byte [esi], 0
5806 00009A77 762D <1> jna short loc_path_retn

```

```

5807 <1>
5808 00009A79 BE[532F0100] <1> mov esi, Cmd_Path ; 'PATH' address
5809 <1> ; 25/07/2022 (*)
5810 00009A7E EB9F <1> jmp short loc_print_env_string
5811 <1> ; 25/07/2022
5812 <1> ; mov edi, TextBuffer ; out buffer
5813 <1> ; xor al, al ; use [ESI]
5814 <1> ; mov ecx, 255 ; maximum size (limit)
5815 <1> ; call get_environment_string
5816 <1> ; jc short loc_path_retn
5817 <1> ; 25/07/2022
5818 <1> ; jnc short loc_print_env_string
5819 <1> ; retn
5820 <1>
5821 <1> ; mov esi, TextBuffer
5822 <1> ; call print_msg
5823 <1> ; mov esi, nextline
5824 <1> ; call print_msg
5825 <1> ; loc_path_retn:
5826 <1> ; retn
5827 <1> ; 25/07/2022
5828 <1> ; jmp print_msg
5829 <1>
5830 <1> loc_set_path:
5831 00009A80 56 <1> push esi
5832 <1> loc_set_path_find_end:
5833 00009A81 46 <1> inc esi
5834 00009A82 803E20 <1> cmp byte [esi], 20h
5835 00009A85 73FA <1> jnb short loc_set_path_find_end
5836 00009A87 C60600 <1> mov byte [esi], 0
5837 <1> loc_set_path_header:
5838 00009A8A 5E <1> pop esi
5839 <1> set_path_x: ; 31/12/2017 ('syspath')
5840 00009A8B 4E <1> dec esi
5841 00009A8C C6063D <1> mov byte [esi], '='
5842 00009A8F 4E <1> dec esi
5843 00009A90 C60648 <1> mov byte [esi], 'H'
5844 00009A93 4E <1> dec esi
5845 00009A94 C60654 <1> mov byte [esi], 'T'
5846 00009A97 4E <1> dec esi
5847 00009A98 C60641 <1> mov byte [esi], 'A'
5848 00009A9B 4E <1> dec esi
5849 00009A9C C60650 <1> mov byte [esi], 'P'
5850 <1>
5851 <1> loc_path_call_set_env_string:
5852 00009A9F E8C6000000 <1> call set_environment_string
5853 00009AA4 72B1 <1> jc short loc_set_cmd_failed
5854 <1> loc_path_retn: ; 25/07/2022
5855 00009AA6 C3 <1> retn
5856 <1>
5857 <1> get_environment_string:
5858 <1> ; 12/04/2016
5859 <1> ; 11/04/2016
5860 <1> ; 05/04/2016 (TRDOS 386 = TRDOS v2.0)
5861 <1> ; 02/09/2011 (TRDOS v1, MAINPROG.ASM)
5862 <1> ; 28/08/2011
5863 <1> ; INPUT->
5864 <1> ; EDI = Output buffer
5865 <1> ; CX = Buffer length (<= ENV_PAGE_SIZE)
5866 <1> ;
5867 <1> ; AL > 0 = AL = String sequence number
5868 <1> ; AL = 0 -> ESI = ASCIIZ Set word
5869 <1> ; (environment variable)
5870 <1> ; OUTPUT ->
5871 <1> ; ESI is not changed
5872 <1> ; EDI is not changed
5873 <1> ; EAX = String length (with zero tail)
5874 <1> ; EDX = Environment variables page address
5875 <1> ; CF = 1 -> Not found (EAX not valid)
5876 <1> ;
5877 <1> ; (Modified registers: EAX, EDX)
5878 <1>
5879 00009AA7 BA00300900 <1> mov edx, Env_Page
5880 00009AAC 803A00 <1> cmp byte [edx], 0
5881 00009AAF 7474 <1> jz short get_env_string_with_word_stc_retn
5882 <1>
5883 00009AB1 66890D[CE820100] <1> mov [env_var_length], cx
5884 <1>
5885 00009AB8 51 <1> push ecx ; *
5886 00009AB9 56 <1> push esi ; **
5887 <1>
5888 00009ABA 08C0 <1> or al, al
5889 00009ABC 7449 <1> jz short get_env_string_with_word
5890 <1>
5891 <1> get_env_string_with_seq_number:
5892 00009ABE 8101 <1> mov cl, 1
5893 00009AC0 88C5 <1> mov ch, al
5894 00009AC2 31C0 <1> xor eax, eax
5895 00009AC4 89D6 <1> mov esi, edx ; Env_Page
5896 <1>
5897 <1> get_env_string_seq_number_check:
5898 00009AC6 38CD <1> cmp ch, cl
5899 00009AC8 7726 <1> ja short get_env_string_seq_number_next
5900 <1>
5901 <1> get_env_string_move_to_buff:
5902 00009ACA 57 <1> push edi ; ***
5903 <1>
5904 00009ACB 29D2 <1> sub edx, edx
5905 <1>
5906 <1> get_env_string_seq_number_repeat1:
5907 00009ACD 42 <1> inc edx
5908 00009ACE AC <1> lodsb
5909 00009ACF AA <1> stosb
5910 <1>
5911 00009AD0 66FF0D[CE820100] <1> dec word [env_var_length]
5912 00009AD7 7508 <1> jnz short get_env_string_seq_number_repeat3
5913 <1>
5914 <1> get_env_string_seq_number_repeat2:
5915 00009AD9 20C0 <1> and al, al
5916 00009ADB 7408 <1> jz short get_env_string_seq_number_ok
5917 00009ADD 42 <1> inc edx
5918 00009ADE AC <1> lodsb
5919 00009ADF EBF8 <1> jmp short get_env_string_seq_number_repeat2
5920 <1>
5921 <1> get_env_string_seq_number_repeat3:
5922 00009AE1 08C0 <1> or al, al
5923 00009AE3 75E8 <1> jnz short get_env_string_seq_number_repeat1
5924 <1>
5925 <1> get_env_string_seq_number_ok:
5926 00009AE5 5F <1> pop edi ; ***
5927 00009AE6 89D0 <1> mov eax, edx ; Length of the environment string
5928 <1> ; (ASCIIZ, includes ZERO tail)
5929 00009AE8 BA00300900 <1> mov edx, Env_Page
5930 <1>

```

```

5931 <1> get_env_string_stc_retn:
5932 00009AED 5E <1> pop esi ; **
5933 00009AEE 59 <1> pop ecx ; *
5934 00009AEF C3 <1> retn
5935 <1>
5936 <1> get_env_string_seq_number_next:
5937 00009AF0 AC <1> lodsb
5938 00009AF1 08C0 <1> or al, al
5939 00009AF3 75FB <1> jnz short get_env_string_seq_number_next
5940 <1>
5941 00009AF5 81FE00320900 <1> cmp esi, Env_Page + Env_Page_Size ; +512 (+4096)
5942 00009AFB F5 <1> cmc
5943 00009AFC 72EF <1> jc short get_env_string_stc_retn
5944 <1>
5945 00009AFE AC <1> lodsb
5946 00009AFF 3C01 <1> cmp al, 1
5947 00009B01 72EA <1> jb short get_env_string_stc_retn
5948 00009B03 FEC1 <1> inc cl
5949 00009B05 EBBF <1> jmp short get_env_string_seq_number_check
5950 <1>
5951 <1> get_env_string_with_word:
5952 00009B07 31C9 <1> xor ecx, ecx
5953 <1>
5954 <1> get_env_string_calc_word_length:
5955 00009B09 AC <1> lodsb
5956 00009B0A 3C20 <1> cmp al, 20h
5957 00009B0C 7211 <1> jb short get_env_string_calc_word_length_ok
5958 <1> ;inc cx
5959 00009B0E FEC1 <1> inc cl
5960 <1>
5961 00009B10 3C61 <1> cmp al, 'a'
5962 00009B12 72F5 <1> jb short get_env_string_calc_word_length
5963 00009B14 3C7A <1> cmp al, 'z'
5964 00009B16 77F1 <1> ja short get_env_string_calc_word_length
5965 00009B18 24DF <1> and al, 0DFh
5966 00009B1A 8846FF <1> mov [esi-1], al
5967 00009B1D EBEA <1> jmp short get_env_string_calc_word_length
5968 <1>
5969 <1> get_env_string_calc_word_length_ok:
5970 00009B1F 08C9 <1> or cl, cl
5971 00009B21 7506 <1> jnz short get_env_string_calc_word_length_save
5972 <1>
5973 00009B23 5E <1> pop esi ; **
5974 <1>
5975 <1> get_env_string_stc_retn1:
5976 00009B24 59 <1> pop ecx ; *
5977 <1>
5978 <1> get_env_string_with_word_stc_retn:
5979 00009B25 31C0 <1> xor eax, eax
5980 00009B27 F9 <1> stc
5981 00009B28 C3 <1> retn
5982 <1>
5983 <1> get_env_string_calc_word_length_save:
5984 00009B29 871C24 <1> xchg ebx, [esp] ; **
5985 00009B2C 89DE <1> mov esi, ebx
5986 <1> ; Start of the env string (to be searched)
5987 <1>
5988 00009B2E 57 <1> push edi ; ***
5989 00009B2F 89D7 <1> mov edi, edx ; Env_Page
5990 <1>
5991 <1> get_env_string_compare:
5992 00009B31 57 <1> push edi ; ****
5993 00009B32 51 <1> push ecx ; ***** ; Variable name length
5994 <1>
5995 <1> get_env_string_compare_rep:
5996 00009B33 AC <1> lodsb
5997 00009B34 AE <1> scasb
5998 00009B35 7511 <1> jne short get_env_string_compare_next1
5999 00009B37 E2FA <1> loop get_env_string_compare_rep
6000 <1>
6001 00009B39 803F3D <1> cmp byte [edi], '='
6002 00009B3C 750A <1> jne short get_env_string_compare_next1
6003 <1>
6004 00009B3E 59 <1> pop ecx ; *****
6005 00009B3F 5F <1> pop edi ; ****
6006 00009B40 89FE <1> mov esi, edi
6007 00009B42 5F <1> pop edi ; ***
6008 00009B43 871C24 <1> xchg ebx, [esp] ; **
6009 00009B46 EB82 <1> jmp short get_env_string_move_to_buff
6010 <1>
6011 <1> get_env_string_compare_next1:
6012 00009B48 89FE <1> mov esi, edi
6013 00009B4A 59 <1> pop ecx ; *****
6014 00009B4B 5F <1> pop edi ; ****
6015 <1>
6016 00009B4C 81FEFF310900 <1> get_env_string_compare_next2:
6017 00009B52 7310 <1> cmp esi, Env_Page + Env_Page_Size - 1 ; +511 (+4095)
6018 00009B54 20C0 <1> jnb short get_env_string_compare_not_ok
6019 00009B56 AC <1> and al, al
6020 00009B57 75F3 <1> lodsb
6021 00009B59 08C0 <1> jnz short get_env_string_compare_next2
6022 00009B5B 7407 <1> or al, al
6023 00009B5D 4E <1> jz short get_env_string_compare_not_ok
6024 00009B5E 89F7 <1> dec esi ; 12/04/2016
6025 00009B60 89DE <1> mov edi, esi
6026 00009B62 EBCD <1> mov esi, ebx
6027 <1> jmp short get_env_string_compare
6028 <1>
6029 00009B64 5F <1> get_env_string_compare_not_ok:
6030 00009B65 89DE <1> pop edi ; ***
6031 00009B67 5B <1> mov esi, ebx
6032 00009B68 EBBA <1> pop ebx ; **
6033 <1> jmp short get_env_string_stc_retn1
6034 <1>
6035 <1> set_environment_string:
6036 <1> ; 25/07/2022 - TRDOS 386 kernel v2.0.5
6037 <1> ; 13/04/2016
6038 <1> ; 12/04/2016
6039 <1> ; 11/04/2016
6040 <1> ; 06/04/2016
6041 <1> ; 05/04/2016 (TRDOS 386 = TRDOS v2.0)
6042 <1> ; 02/09/2011 (TRDOS v1, MAINPROG.ASM)
6043 <1> ; 29/08/2011
6044 <1> ; 29/08/2011
6045 <1> ; INPUT->
6046 <1> ; ESI = ASCIIIZ environment string
6047 <1> ; OUTPUT ->
6048 <1> ; ESI is not changed
6049 <1> ; CF = 1 -> Could not set,
6050 <1> ; insufficient environment space
6051 <1> ;
6052 <1> ; (EAX, EDX will be changed)
6053 <1> ;
6054 <1> ; (EAX = Start address of the env string if > 0)
6055 <1> ; (EDX = Environment string length)

```

```

6055 <1>
6056 00009B6A 56 <1> push esi ; *
6057 <1>
6058 00009B6B 31C0 <1> xor eax, eax
6059 <1>
6060 <1> set_env_chk_validation1:
6061 00009B6D FEC4 <1> inc ah ; variable (string) length
6062 00009B6F AC <1> lodsb
6063 00009B70 3C3D <1> cmp al, '='
6064 00009B72 7415 <1> je short set_env_chk_validation2
6065 00009B74 3C20 <1> cmp al, 20h
6066 00009B76 720F <1> jnb short set_env_string_stc
6067 <1>
6068 <1> ; 06/04/2016
6069 00009B78 3C61 <1> cmp al, 'a'
6070 00009B7A 72F1 <1> jnb short set_env_chk_validation1
6071 00009B7C 3C7A <1> cmp al, 'z'
6072 00009B7E 77ED <1> ja short set_env_chk_validation1
6073 00009B80 2C20 <1> sub al, 'a'-'A'
6074 00009B82 8846FF <1> mov [esi-1], al
6075 00009B85 EBE6 <1> jmp short set_env_chk_validation1
6076 <1>
6077 <1> set_env_string_stc:
6078 00009B87 5E <1> pop esi ; *
6079 <1> ;stc
6080 00009B88 C3 <1> retn
6081 <1>
6082 <1> set_env_chk_validation2:
6083 00009B89 51 <1> push ecx ; **
6084 00009B8A 53 <1> push ebx ; ***
6085 00009B8B 57 <1> push edi ; ****
6086 <1>
6087 <1> ; 12/04/2016
6088 <1> ;mov ebx, [esp+12]
6089 <1> ; 25/07/2022
6090 00009B8C 8B54240C <1> mov edx, [esp+12]
6091 <1>
6092 <1> set_env_chk_validation2w:
6093 00009B90 89F7 <1> mov edi, esi
6094 00009B92 4F <1> dec edi
6095 <1>
6096 00009B93 807FFF20 <1> cmp byte [edi-1], 20h
6097 00009B97 771A <1> ja short set_env_chk_validation2z
6098 <1>
6099 00009B99 56 <1> push esi
6100 00009B9A 89FE <1> mov esi, edi
6101 00009B9C 4E <1> dec esi
6102 <1>
6103 <1> set_env_chk_validation2x:
6104 00009B9D 4E <1> dec esi
6105 <1>
6106 <1> ;cmp esi, ebx
6107 00009B9E 39D6 <1> cmp esi, edx ; 25/07/2022
6108 00009BA0 7207 <1> jnb short set_env_chk_validation2y
6109 <1>
6110 00009BA2 4F <1> dec edi
6111 <1>
6112 00009BA3 8A06 <1> mov al, [esi]
6113 00009BA5 8807 <1> mov [edi], al
6114 <1>
6115 00009BA7 EBF4 <1> jmp short set_env_chk_validation2x
6116 <1>
6117 <1> set_env_chk_validation2y:
6118 00009BA9 5E <1> pop esi
6119 <1>
6120 <1> ;mov byte [ebx], 20h
6121 <1> ; 25/07/2022
6122 <1> ;mov byte [edx], 20h
6123 <1>
6124 <1> ;inc ebx
6125 <1> ;mov [esp+12], ebx
6126 <1> ; 25/07/2022
6127 00009BAA 42 <1> inc edx
6128 00009BAB 8954240C <1> mov [esp+12], edx
6129 <1>
6130 00009BAF FECC <1> dec ah ; 13/04/2016
6131 <1>
6132 00009BB1 EBDD <1> jmp short set_env_chk_validation2w
6133 <1>
6134 <1> set_env_chk_validation2z:
6135 <1> ;mov edx, Env_Page
6136 <1> ;mov edi, edx
6137 <1> ; 25/07/2022
6138 00009BB3 BB00300900 <1> mov ebx, Env_Page
6139 00009BB8 89DF <1> mov edi, ebx
6140 <1>
6141 <1> set_env_chk_validation3:
6142 00009BBA AC <1> lodsb
6143 00009BBB 3C20 <1> cmp al, 20h
6144 00009BBD 74FB <1> je short set_env_chk_validation3
6145 <1>
6146 00009BBF 9C <1> pushf
6147 <1>
6148 <1> ; 12/04/2016
6149 <1> set_env_chk_validation3n:
6150 00009BC0 3C61 <1> cmp al, 'a'
6151 00009BC2 720C <1> jnb short set_env_chk_validation3c
6152 00009BC4 3C7A <1> cmp al, 'z'
6153 00009BC6 7705 <1> ja short set_env_chk_validation3x
6154 00009BC8 2C20 <1> sub al, 'a'-'A'
6155 00009BCA 8846FF <1> mov [esi-1], al
6156 <1>
6157 <1> set_env_chk_validation3x:
6158 00009BCD AC <1> lodsb
6159 00009BCE EBF0 <1> jmp short set_env_chk_validation3n
6160 <1>
6161 <1> set_env_chk_validation3c:
6162 00009BD0 3C20 <1> cmp al, 20h
6163 00009BD2 73F9 <1> jnb short set_env_chk_validation3x
6164 <1>
6165 00009BD4 803F00 <1> cmp byte [edi], 0
6166 00009BD7 772B <1> ja short set_env_chk_validation4
6167 <1>
6168 00009BD9 9D <1> popf
6169 00009BDA 7222 <1> jnb short set_env_string_nothing
6170 <1>
6171 00009BDC B900020000 <1> mov ecx, Env_Page_Size ; 512 (4096)
6172 <1>
6173 <1> ;mov esi, ebx ; 12/04/2016
6174 <1> ; 25/07/2022
6175 00009BE1 89D6 <1> mov esi, edx
6176 <1>
6177 <1> ; 25/07/2022
6178 00009BE3 89CA <1> mov edx, ecx

```

```

6179 <1>
6180 <1> set_env_string_copy_to_envb:
6181 <1> lodsb
6182 <1> cmp al, 20h
6183 <1> jb short set_env_string_copy_to_envb_z
6184 <1> stosb
6185 <1> loop set_env_string_copy_to_envb
6186 <1>
6187 <1> ; 11/04/2016
6188 <1> ;mov edi, edx ; Env_Page
6189 <1> ; 25/07/2022
6190 <1> mov edi, ebx
6191 <1> ; 25/07/2022
6192 <1> ;mov ecx, Env_Page_Size
6193 <1> mov ecx, edx
6194 <1>
6195 <1> set_env_string_copy_to_envb_z:
6196 <1> ; 25/07/2022
6197 <1> ;push edx ; Start address of the variable
6198 <1>
6199 <1> ;mov edx, Env_Page_Size
6200 <1> ; 25/07/2022
6201 <1> sub edx, ecx ; variable (string) length
6202 <1>
6203 <1> sub al, al ; 0
6204 <1> rep stosb ; clear remain bytes of the env page
6205 <1>
6206 <1> ;pop eax ; Start address of the variable
6207 <1> ; 25/07/2022
6208 <1> mov eax, ebx
6209 <1>
6210 <1> set_env_string_allocate_envb_retn: ; stc or clc return
6211 <1> pop edi ; ****
6212 <1> pop ebx ; ***
6213 <1> pop ecx ; **
6214 <1> pop esi ; *
6215 <1> retn
6216 <1>
6217 <1> set_env_string_nothing:
6218 <1> xor eax, eax
6219 <1> xor edx, edx ; 11/04/2016
6220 <1> jmp short set_env_string_allocate_envb_retn
6221 <1>
6222 <1> set_env_chk_validation4:
6223 <1> ; 11/04/2016
6224 <1> popf
6225 <1>
6226 <1> ;mov esi, edx ; Env_Page
6227 <1> ; 25/07/2022
6228 <1> mov esi, ebx
6229 <1>
6230 <1> set_env_chk_validation5:
6231 <1> ;mov edi, ebx ; ASCIIZ environment string address
6232 <1> ; 25/07/2022
6233 <1> mov edi, edx
6234 <1> movzx ecx, ah ; variable (string) length (with '=')
6235 <1>
6236 <1> set_env_chk_validation5_loop:
6237 <1> lodsb
6238 <1> scasb
6239 <1> jne short set_env_chk_validation6
6240 <1> loop set_env_chk_validation5_loop
6241 <1>
6242 <1> cmp al, '='
6243 <1> ;je set_env_change_variable
6244 <1> ; 25/07/2022
6245 <1> jne short set_env_chk_validation6
6246 <1> jmp set_env_change_variable
6247 <1>
6248 <1> set_env_chk_validation6:
6249 <1> or al, al ; 0
6250 <1> jz short set_env_chk_validation7
6251 <1>
6252 <1> lodsb
6253 <1> jmp short set_env_chk_validation6
6254 <1>
6255 <1> set_env_chk_validation7:
6256 <1> mov cl, ah
6257 <1> add ecx, esi
6258 <1> cmp ecx, Env_Page + Env_Page_Size - 1
6259 <1> ; 511 (4095)
6260 <1> ; strlen + '=' + 0
6261 <1> jb short set_env_chk_validation5
6262 <1>
6263 <1> set_env_chk_validation8: ; variable not found
6264 <1> movzx esi, ah ; variable name length (with '=')
6265 <1> ;add esi, ebx ; position just after of the '='
6266 <1> ; 25/07/2022
6267 <1> add esi, edx
6268 <1>
6269 <1> set_env_chk_validation8_loop:
6270 <1> lodsb
6271 <1> cmp al, 20h
6272 <1> je short set_env_chk_validation8_loop
6273 <1> jb short set_env_string_nothing
6274 <1>
6275 <1> set_env_chk_validation9:
6276 <1> lodsb
6277 <1> cmp al, 20h
6278 <1> jnb short set_env_chk_validation9
6279 <1>
6280 <1> ; End of ASCIIZ environment string
6281 <1>
6282 <1> set_env_add_variable:
6283 <1> ;sub esi, ebx ; variable+definition length
6284 <1> ; 25/07/2022
6285 <1> sub esi, edx
6286 <1>
6287 <1> push esi ; *****
6288 <1>
6289 <1> ;mov esi, edx ; Environment page address
6290 <1> ; 25/07/2022
6291 <1> mov esi, ebx
6292 <1>
6293 <1> mov ecx, Env_Page_Size ; 512 (4096)
6294 <1>
6295 <1> set_env_add_variable_loop:
6296 <1> lodsb
6297 <1> and al, al
6298 <1> jz short set_env_add_variable_chk1 ; 0
6299 <1> loop set_env_add_variable_loop
6300 <1>
6301 <1> ; 11/04/2016
6302 <1> mov [esi-1], cl ; 0

```

```

6303 00009C50 41      <1>      inc      ecx
6304                <1>
6305                <1> set_env_add_variable_chk1:
6306 00009C51 49      <1>      dec      ecx
6307 00009C52 7408    <1>      jz       short set_env_add_variable_nspc
6308 00009C54 AC      <1>      lodsb
6309 00009C55 08C0    <1>      or       al, al
6310 00009C57 740B    <1>      jz       short set_env_add_variable_chk2 ; 00
6311 00009C59 49      <1>      dec      ecx
6312 00009C5A 75EA    <1>      jnz      short set_env_add_variable_loop
6313                <1>
6314                <1> set_env_add_variable_nspc: ; no space on environment page
6315 00009C5C 58      <1>      pop      eax ; *****
6316                <1>      ;mov     eax, 8 ; No space for new environment string
6317                <1>      ; 25/07/2022
6318 00009C5D 29C0    <1>      sub      eax, eax
6319 00009C5F B008    <1>      mov      al, 8
6320 00009C61 F9      <1>      stc
6321 00009C62 EB95    <1>      jmp      short set_env_string_allocate_envb_retn
6322                <1>
6323                <1> set_env_add_variable_chk2:
6324 00009C64 880C24  <1>      mov      ecx, [esp] ; *****
6325 00009C67 4E      <1>      dec      esi ; beginning address of the new variable
6326 00009C68 89F0    <1>      mov      eax, esi
6327 00009C6A 01C8    <1>      add      eax, ecx ; string length (with CR)
6328                <1>      ;add     edx, Env_Page_Size ; 512 (4096)
6329                <1>      ; 25/07/2022
6330 00009C6C 81C300020000 <1>      add      ebx, Env_Page_Size
6331                <1>      ;cmp     eax, edx
6332 00009C72 39D8    <1>      cmp      eax, ebx ; 25/07/2022
6333 00009C74 77E6    <1>      ja       short set_env_add_variable_nspc
6334 00009C76 49      <1>      dec      ecx ; except CR at the end
6335                <1>      ;mov     edx, ecx ; 12/04/2016
6336                <1>      ; 25/07/2022
6337 00009C77 89CB    <1>      mov      ebx, ecx
6338 00009C79 89F7    <1>      mov      edi, esi
6339 00009C7B 893C24  <1>      mov      [esp], edi ; ***** ; Start address of new variable
6340                <1>      ;mov     esi, ebx ; ASCIIZ environment string address
6341                <1>      ; 25/07/2022
6342 00009C7E 89D6    <1>      mov      esi, edx
6343 00009C80 F3A4    <1>      rep      movsb
6344 00009C82 28C0    <1>      sub      al, al
6345 00009C84 AA      <1>      stosb
6346 00009C85 58      <1>      pop      eax ; ***** ; Beginning address of new variable
6347 00009C86 81FF00320900 <1>      cmp      edi, Env_Page + Env_Page_Size ; 12/04/2016
6348                <1>      ;jnb     set_env_string_allocate_envb_retn ; OK !
6349                <1>      ; 25/07/2022
6350 00009C8C 7303    <1>      jnb      short set_env_add_variable_chk3
6351 00009C8E 880F    <1>      mov      [edi], cl ; 0
6352 00009C90 F8      <1>      cld      ; 13/04/2016
6353                <1> set_env_add_variable_chk3:
6354 00009C91 E963FFFFFF <1>      jmp      set_env_string_allocate_envb_retn ; OK !
6355                <1>
6356                <1> set_env_change_variable:
6357                <1>      ; 06/04/2016
6358                <1>      ; esi = Variable's address in environment page (after '=')
6359                <1>      ; edi = ASCIIZ environment string address (after '=')
6360                <1>
6361                <1>      ; ah = variable length from start to the '='
6362 00009C96 8825[CE820100] <1>      mov      [env_var_length], ah
6363                <1>
6364 00009C9C 28C9    <1>      sub      cl, cl ; ecx = 0
6365                <1>
6366 00009C9E 57      <1>      push     edi ; *****
6367                <1>
6368 00009C9F 89F7    <1>      mov      edi, esi ; 11/04/2016
6369                <1>
6370                <1> set_env_change_variable_calc1:
6371 00009CA1 AC      <1>      lodsb
6372 00009CA2 08C0    <1>      or       al, al
6373 00009CA4 7403    <1>      jz       short set_env_change_variable_calc2
6374                <1>
6375 00009CA6 41      <1>      inc      ecx ; length of environment string (after the '=')
6376                <1>
6377 00009CA7 EBF8    <1>      jmp      short set_env_change_variable_calc1
6378                <1>
6379                <1> set_env_change_variable_calc2:
6380 00009CA9 8B3424  <1>      mov      esi, [esp] ; ASCIIZ environment string address
6381                <1>
6382 00009CAC 29D2    <1>      sub      edx, edx
6383                <1>
6384                <1> set_env_change_variable_calc3:
6385 00009CAE AC      <1>      lodsb
6386 00009CAF 3C20    <1>      cmp      al, 20h
6387 00009CB1 7203    <1>      jb       short set_env_change_variable_calc4
6388                <1>
6389 00009CB3 42      <1>      inc      edx ; length of ASCIIZ string (after the '=')
6390                <1>
6391 00009CB4 EBF8    <1>      jmp      short set_env_change_variable_calc3
6392                <1>
6393                <1> set_env_change_variable_calc4:
6394 00009CB6 C646FF00 <1>      mov      byte [esi-1], 0 ; put ZERO instead of CR
6395                <1>
6396 00009CBA 5E      <1>      pop      esi ; ***** ; ASCIIZ string address (after '=')
6397                <1>
6398                <1>      ; EDI = Old variable's address (after '=')
6399                <1>
6400                <1>      ; compare the new string with the old string
6401 00009CBB 39CA    <1>      cmp      edx, ecx
6402 00009CBD 7718    <1>      ja       short set_env_change_variable_calc5 ; longer
6403                <1>      ;jb      set_env_change_variable_calc9 ; shorter
6404                <1>      ; 25/07/2022
6405 00009CBF 7405    <1>      je       short set_env_change_variable_calc22
6406 00009CC1 E98C000000 <1>      jmp      set_env_change_variable_calc9
6407                <1>
6408                <1> set_env_change_variable_calc22:
6409                <1>      ;same length (simple copy)
6410 00009CC6 0FB6C4  <1>      movzx    eax, ah
6411 00009CC9 01C2    <1>      add      edx, eax
6412 00009CCB F7D8    <1>      neg      eax
6413 00009CCD 01F8    <1>      add      eax, edi
6414                <1>      ; EAX = Start address of the variable
6415                <1>      ; EDX = Variable length (without ZERO at the end of variable)
6416                <1>
6417 00009CCF F3A4    <1>      rep      movsb
6418 00009CD1 F8      <1>      cld      ; 13/04/2016
6419 00009CD2 E922FFFFFF <1>      jmp      set_env_string_allocate_envb_retn ; OK !
6420                <1>
6421                <1> set_env_change_variable_calc5:
6422                <1>      ; 11/04/2016
6423 00009CD7 52      <1>      push     edx ; *****
6424 00009CD8 29CA    <1>      sub      edx, ecx ; difference ; (the new string is longer)
6425 00009CDA 89F3    <1>      mov      ebx, esi
6426 00009CDC 89FE    <1>      mov      esi, edi

```

```

6427 <1>
6428 <1> set_env_change_variable_calc6:
6429 <1> lodsb
6430 <1> and al, al
6431 <1> jnz short set_env_change_variable_calc6
6432 <1>
6433 <1> cmp esi, Env_Page + Env_Page_Size ; 512 (4096)
6434 <1> jnb set_env_add_variable_nspc
6435 <1>
6436 <1> mov ecx, edi ; current (old) variable's address
6437 <1> mov edi, esi ; next variable's address
6438 <1>
6439 <1> lodsb
6440 <1> or al, al
6441 <1> jz short set_env_change_variable_calc8 ; 00
6442 <1>
6443 <1> set_env_change_variable_calc7:
6444 <1> lodsb
6445 <1> and al, al
6446 <1> jnz short set_env_change_variable_calc7
6447 <1>
6448 <1> cmp esi, Env_Page + Env_Page_Size ; 512 (4096)
6449 <1> ;jnb set_env_add_variable_nspc
6450 <1> ; 25/07/2022
6451 <1> ;b short set_env_change_variable_calc24
6452 <1> set_env_change_variable_calc23:
6453 <1> jmp set_env_add_variable_nspc
6454 <1>
6455 <1> set_env_change_variable_calc24:
6456 <1> lodsb
6457 <1> or al, al
6458 <1> jnz short set_env_change_variable_calc7
6459 <1>
6460 <1> set_env_change_variable_calc8:
6461 <1> dec esi ; address of the second (last) 0 of the 00
6462 <1>
6463 <1> add edx, esi ; final position of the last 0
6464 <1>
6465 <1> cmp edx, Env_Page + Env_Page_Size ; 512 (4096)
6466 <1> ;jnb set_env_add_variable_nspc
6467 <1> ; 25/07/2022
6468 <1> jnb short set_env_change_variable_calc23
6469 <1>
6470 <1> mov eax, ecx ; old variable's address (after '=')
6471 <1>
6472 <1> mov ecx, esi
6473 <1> sub ecx, edi ; count of bytes to move forward
6474 <1>
6475 <1> ; 13/04/2016
6476 <1> mov byte [edx], 0
6477 <1> mov edi, edx
6478 <1> sub edx, esi ; difference (additional byte count)
6479 <1> dec edi ; the last zero address (first byte of the 00)
6480 <1> mov esi, edi
6481 <1> sub esi, edx ; - displacement
6482 <1>
6483 <1> cli ; disable interrupts
6484 <1> std ; backward
6485 <1>
6486 <1> rep movsb ; move ECX bytes from DS:ESI to ES:EDI
6487 <1>
6488 <1> cld ; forward (default)
6489 <1> sti ; enable interrupts
6490 <1>
6491 <1> mov edi, eax
6492 <1> pop ecx ; ***** ; byte count (after '=')
6493 <1> mov edx, ecx
6494 <1> mov esi, ebx ; ASCIIIZ string address (after '=')
6495 <1> mov ebx, edi
6496 <1>
6497 <1> rep movsb
6498 <1>
6499 <1> mov [edi], cl ; 0 ; end of variable
6500 <1>
6501 <1> movzx eax, byte [env_var_length]
6502 <1> add edx, eax ; variable length (total)
6503 <1> neg eax
6504 <1> add eax, ebx ; start address of the variable
6505 <1> clc ; 13/04/2016
6506 <1> jmp set_env_string_allocate_envb_retn ; OK !
6507 <1>
6508 <1> set_env_change_variable_calc9:
6509 <1> ; 11/04/2016
6510 <1> and edx, edx ; is empty ?
6511 <1> jnz short set_env_change_variable_calc15
6512 <1>
6513 <1> movzx ebx, ah
6514 <1> neg ebx
6515 <1> add ebx, edi
6516 <1>
6517 <1> ; EBX = Start address of the variable (in env page)
6518 <1> ; EDX = Variable length = 0
6519 <1>
6520 <1> mov esi, edi
6521 <1>
6522 <1> set_env_change_variable_calc10:
6523 <1> lodsb
6524 <1> or al, al
6525 <1> jnz short set_env_change_variable_calc10
6526 <1>
6527 <1> mov ecx, Env_Page + Env_Page_Size - 1
6528 <1>
6529 <1> cmp esi, ecx ; +511 (+4095)
6530 <1> jna short set_env_change_variable_calc11
6531 <1>
6532 <1> mov esi, ecx
6533 <1> mov [esi], al ; 0
6534 <1>
6535 <1> set_env_change_variable_calc11:
6536 <1> mov edi, ebx ; old variable's start address
6537 <1>
6538 <1> set_env_change_variable_calc12:
6539 <1> lodsb
6540 <1> stosb
6541 <1> and al, al
6542 <1> jnz short set_env_change_variable_calc12
6543 <1> cmp esi, ecx
6544 <1> ja short set_env_change_variable_calc13
6545 <1> lodsb
6546 <1> stosb
6547 <1> and al, al
6548 <1> jnz short set_env_change_variable_calc12
6549 <1>
6550 <1> set_env_change_variable_calc13:

```

```

6551 00009D83 29F9      <1>      sub     ecx, edi
6552 00009D85 7203      <1>      jb      short set_env_change_variable_calc14
6553 00009D87 41         <1>      inc     ecx ; 1-512 (1-4096)
6554 00009D88 F3AA      <1>      rep     stosb ; al = 0
6555                                     <1>
6556                                     <1> set_env_change_variable_calc14:
6557 00009D8A 29C0      <1>      sub     eax, eax ; Start address of the variable
6558                                     <1>      ; EAX = 0 -> variable is removed
6559                                     <1>      ; EDX = variable length = 0
6560                                     <1>
6561 00009D8C E968FEFFFF <1>      jmp     set_env_string_allocate_envb_retn ; OK !
6562                                     <1>
6563                                     <1> set_env_change_variable_calc15:
6564 00009D91 52         <1>      push    edx ; *****
6565 00009D92 F7DA      <1>      neg     edx
6566 00009D94 01CA      <1>      add     edx, ecx ; difference (the old string is longer)
6567 00009D96 89F3      <1>      mov     ebx, esi
6568 00009D98 89FE      <1>      mov     esi, edi
6569                                     <1>
6570                                     <1> set_env_change_variable_calc16:
6571 00009D9A AC         <1>      lodsb
6572 00009D9B 20C0      <1>      and     al, al
6573 00009D9D 75FB      <1>      jnz     short set_env_change_variable_calc16
6574                                     <1>
6575 00009D9F B900320900 <1>      mov     ecx, Env_Page + Env_Page_Size
6576                                     <1>
6577 00009DA4 39CE      <1>      cmp     esi, ecx ; +512 (+4096)
6578 00009DA6 7605      <1>      jna     short set_env_change_variable_calc17
6579                                     <1>
6580 00009DA8 89CE      <1>      mov     esi, ecx
6581 00009DAA 8846FF <1>      mov     [esi-1], al ; 0
6582                                     <1>
6583                                     <1> set_env_change_variable_calc17:
6584 00009DAD 89F9      <1>      mov     ecx, edi ; current (old) variable's address
6585 00009DAF 89F7      <1>      mov     edi, esi ; next variable's address
6586                                     <1>
6587 00009DB1 AC         <1>      lodsb
6588 00009DB2 08C0      <1>      or      al, al
6589 00009DB4 741D      <1>      jz      short set_env_change_variable_calc20
6590                                     <1>
6591                                     <1> set_env_change_variable_calc18:
6592 00009DB6 AC         <1>      lodsb
6593 00009DB7 20C0      <1>      and     al, al
6594 00009DB9 75FB      <1>      jnz     short set_env_change_variable_calc18
6595                                     <1>
6596 00009DBB 81FE00320900 <1>      cmp     esi, Env_Page + Env_Page_Size
6597 00009DC1 720B      <1>      jb      short set_env_change_variable_calc19
6598 00009DC3 740E      <1>      je      short set_env_change_variable_calc20
6599                                     <1>
6600 00009DC5 BEFF310900 <1>      mov     esi, Env_Page + Env_Page_Size - 1
6601 00009DCA 8806      <1>      mov     [esi], al ; 0
6602 00009DCC EB06      <1>      jmp     short set_env_change_variable_calc21
6603                                     <1>
6604                                     <1> set_env_change_variable_calc19:
6605 00009DCE AC         <1>      lodsb
6606 00009DCF 08C0      <1>      or      al, al
6607 00009DD1 75E3      <1>      jnz     short set_env_change_variable_calc18
6608                                     <1>
6609                                     <1> set_env_change_variable_calc20:
6610 00009DD3 4E         <1>      dec     esi ; address of the second (last) 0 of the 00
6611                                     <1>
6612                                     <1> set_env_change_variable_calc21:
6613                                     <1>      ; edx = difference (byte count)
6614                                     <1>
6615 00009DD4 89C8      <1>      mov     eax, ecx ; old variable's address (after '=')
6616                                     <1>
6617 00009DD6 89F1      <1>      mov     ecx, esi
6618 00009DD8 29F9      <1>      sub     ecx, edi ; count of bytes to move backward
6619                                     <1>
6620 00009DDA 89FE      <1>      mov     esi, edi ; next variable's address
6621 00009DDC 29D7      <1>      sub     edi, edx ; (displacement)
6622                                     <1>
6623 00009DDE F3A4      <1>      rep     movsb
6624                                     <1>
6625 00009DE0 880F      <1>      mov     [edi], cl ; 0 ; 00 ; end of environment variables
6626                                     <1>
6627 00009DE2 89C7      <1>      mov     edi, eax
6628 00009DE4 5A         <1>      pop     edx ; ***** ; byte count (after '=')
6629 00009DE5 89D1      <1>      mov     ecx, edx
6630 00009DE7 89DE      <1>      mov     esi, ebx ; ASCIIIZ string address (after '=')
6631 00009DE9 89FB      <1>      mov     ebx, edi
6632                                     <1>
6633 00009DEB F3A4      <1>      rep     movsb
6634                                     <1>
6635 00009DED 880F      <1>      mov     [edi], cl ; 0 ; end of variable
6636                                     <1>
6637 00009DEF 0FB605[CE820100] <1>      movzx   eax, byte [env_var_length]
6638 00009DF6 01C2      <1>      add     edx, eax ; variable length (total)
6639 00009DF8 F7D8      <1>      neg     eax
6640 00009DFA 01D8      <1>      add     eax, ebx ; start address of the variable
6641 00009DFC F8         <1>      clc     ; 13/04/2016
6642 00009DFD E9F7FDFFFF <1>      jmp     set_env_string_allocate_envb_retn ; OK !
6643                                     <1>
6644                                     <1> ; 19/12/2025
6645                                     <1> loc_load_mainprog_cfg_exit:
6646 00009E02 C3         <1>      retn
6647                                     <1>
6648                                     <1> mainprog_startup_configuration:
6649                                     <1>      ; 19/12/2025 - TRDOS 386 Kernel v2.10.10
6650                                     <1>      ; 25/07/2022 - TRDOS 386 Kernel v2.0.5
6651                                     <1>      ; 22/11/2017
6652                                     <1>      ; 06/05/2016
6653                                     <1>      ; 14/04/2016 (TRDOS 386 = TRDOS v2.0)
6654                                     <1>      ; 17/09/2011 (TRDOS v1, MAINPROG.ASM)
6655                                     <1>
6656                                     <1> loc_load_mainprog_cfg_file:
6657 00009E03 BE[CD2E0100] <1>      mov     esi, MainProgCfgFile
6658 00009E08 66B80018 <1>      mov     ax, 1800h ; Except volume label and dirs
6659 00009E0C E859EBFFFF <1>      call    find_first_file
6660 00009E11 72EF      <1>      jc      short loc_load_mainprog_cfg_exit
6661                                     <1>
6662                                     <1> ;or     eax, eax
6663                                     <1> ;jz     short loc_load_mainprog_cfg_exit
6664                                     <1>
6665                                     <1> loc_start_mainprog_configuration:
6666                                     <1>      ; ESI = FindFile_DirEntry Location
6667                                     <1>      ; EAX = File Size
6668                                     <1>
6669 00009E13 A3[50770100] <1>      mov     [MainProgCfg_FileSize], eax
6670                                     <1>
6671 00009E18 66B85614 <1>      mov     dx, [esi+DirEntry_FstClusHI]
6672 00009E1C C1E210 <1>      shl     edx, 16
6673 00009E1F 66B8561A <1>      mov     dx, [esi+DirEntry_FstClusLO]
6674                                     <1> ;mov     [csftdf_sf_cluster], edx

```



```

6675 ; 19/12/2025
6676 00009E23 8915[84820100] <1> mov [MainProgCfg_cluster], edx
6677 <1>
6678 00009E29 89C1 <1> mov ecx, eax
6679 00009E2B 29C0 <1> sub eax, eax
6680 <1>
6681 ; TRDOS 386 (TRDOS v2.0)
6682 ; Allocate contiguous memory block for loading the file
6683 <1>
6684 ; eax = 0 (Allocate memory from the beginning)
6685 ; ecx = File (Allocation) size in bytes
6686 <1>
6687 00009E2D E807BFFFFF <1> call allocate_memory_block
6688 00009E32 72CE <1> jc short loc_load_mainprog_cfg_exit
6689 <1>
6690 ;mov [csftdf_sf_mem_addr], eax ; loading address
6691 ;mov [csftdf_sf_mem_bsize], ecx ; block size
6692 ; 19/12/2025
6693 00009E34 A3[7C820100] <1> mov [MainProgCfg_mem_addr], eax ; loading address
6694 00009E39 890D[80820100] <1> mov [MainProgCfg_mem_bsize], ecx ; block size
6695 <1>
6696 00009E3F 31DB <1> xor ebx, ebx
6697 <1> ;mov [csftdf_sf_rbytes], ebx ; 0, reset
6698 ; 19/12/2025
6699 <1> ;mov [MainProgCfg_rbytes], ebx ; 0, reset
6700 <1>
6701 00009E41 8A3D[62770100] <1> mov bh, [Current_Drv] ; [FindFile_Drv]
6702 00009E47 BE00010900 <1> mov esi, Logical_DOSDisks
6703 00009E4C 01DE <1> add esi, ebx
6704 <1>
6705 ;mov ebx, [csftdf_sf_mem_addr] ; memory block address
6706 ; 19/12/2025
6707 00009E4E 8B1D[7C820100] <1> mov ebx, [MainProgCfg_mem_addr]
6708 <1>
6709 ; 19/12/2025
6710 <1> %if 0
6711 <1> cmp byte [esi+LD_FATType], 0
6712 <1> ja short loc_mcfg_load_fat_file
6713 <1>
6714 ;mov dword [csftdf_r_size], 65536
6715 ; 19/12/2025
6716 <1> mov dword [MainProgCfg_r_size], 65536
6717 <1> jmp loc_mcfg_load_fs_file
6718 <1>
6719 loc_load_mainprog_cfg_exit:
6720 <1> retn
6721 <1> %endif
6722 <1>
6723 loc_mcfg_load_fat_file:
6724 <1> ;movzx eax, word [esi+LD_BPB+BytesPerSec]
6725 ; 19/12/2025
6726 00009E54 B800020000 <1> mov eax, 512
6727 00009E59 0FB64E13 <1> movzx ecx, byte [esi+LD_BPB+SecPerClust]
6728 00009E5D F7E1 <1> mul ecx
6729 <1> ;mov [csftdf_r_size], eax
6730 ; 19/12/2025
6731 00009E5F A3[8C820100] <1> mov [MainProgCfg_r_size], eax
6732 <1>
6733 loc_mcfg_load_fat_file_next:
6734 00009E64 E817010000 <1> call mcfg_read_fat_file_sectors
6735 00009E69 7259 <1> jc short mcfg_deallocate_mem ; 25/07/2022
6736 <1>
6737 00009E6B 09D2 <1> or edx, edx ; edx > 0 -> EOF
6738 00009E6D 74F5 <1> jz short loc_mcfg_load_fat_file_next
6739 <1>
6740 loc_mcfg_load_fat_file_ok:
6741 ; 06/05/2016
6742 00009E6F C705[20830100]- <1> mov dword [mainprog_return_addr], loc_mcfg_ci_return_addr
6743 00009E75 [419F0000] <1>
6744 <1>
6745 ;mov esi, [csftdf_sf_mem_addr]
6746 ; 19/12/2025
6747 00009E79 8B35[7C820100] <1> mov esi, [MainProgCfg_mem_addr]
6748 00009E7F 8935[54770100] <1> mov [MainProgCfg_LineOffset], esi
6749 <1>
6750 00009E85 A1[50770100] <1> mov eax, [MainProgCfg_FileSize]
6751 00009E8A 89C2 <1> mov edx, eax
6752 00009E8C 01F2 <1> add edx, esi
6753 <1>
6754 00009E8E 89C1 <1> loc_mcfg_process_next_line_check:
6755 <1> mov ecx, eax
6756 <1>
6757 00009E90 803E2A <1> cmp byte [esi], "*" ; Remark sign
6758 00009E93 7503 <1> jne short loc_mcfg_process_next_line
6759 00009E95 46 <1> inc esi
6760 00009E96 EB16 <1> jmp short loc_move_mainprog_cfg_n11
6761 <1>
6762 00009E98 83F94F <1> loc_mcfg_process_next_line:
6763 00009E9B 7604 <1> cmp ecx, 79
6764 <1> jna short loc_start_mainprog_cfg_process
6765 <1>
6766 ;mov ecx, 79
6767 ; 25/07/2022
6768 00009E9D 29C9 <1> sub ecx, ecx
6769 00009E9F B14F <1> mov cl, 79
6770 <1>
6771 00009EA1 BF[12780100] <1> loc_start_mainprog_cfg_process:
6772 <1> mov edi, CommandBuffer
6773 <1>
6774 00009EA6 AC <1> loc_move_mainprog_cfg_line:
6775 00009EA7 3C20 <1> lodsb
6776 00009EA9 720C <1> cmp al, 20h
6777 00009EAB AA <1> jb short loc_move_mainprog_cfg_n12
6778 00009EAC E2F8 <1> stosb
6779 <1> loop loc_move_mainprog_cfg_line
6780 <1>
6781 00009EAE 39D6 <1> loc_move_mainprog_cfg_n11:
6782 00009EB0 7322 <1> cmp esi, edx ; + configuration file size
6783 00009EB2 AC <1> jnb short loc_end_of_mainprog_cfg_line
6784 00009EB3 3C20 <1> lodsb
6785 00009EB5 73F7 <1> cmp al, 20h
6786 <1> jnb short loc_move_mainprog_cfg_n11
6787 <1>
6788 00009EB7 39D6 <1> loc_move_mainprog_cfg_n12:
6789 00009EB9 7319 <1> cmp esi, edx
6790 00009EBB 8A06 <1> jnb short loc_end_of_mainprog_cfg_line
6791 00009EBD 3C20 <1> mov al, [esi]
6792 00009EBF 7713 <1> cmp al, 20h
6793 00009EC1 46 <1> ja short loc_end_of_mainprog_cfg_line
6794 00009EC2 EBF3 <1> inc esi
6795 <1> jmp short loc_move_mainprog_cfg_n12
6796 <1>
6797 ; 25/07/2022
<1> mcfg_deallocate_mem:

```

```

6798      <1>      ;mov     eax, [csftdf_sf_mem_addr] ; start address
6799      <1>      ;mov     ecx, [csftdf_sf_mem_bsize] ; block size
6800      <1>      ; 19/12/2025
6801 00009EC4 A1[7C820100] <1>      mov     eax, [MainProgCfg_mem_addr] ; start address
6802 00009EC9 8B0D[80820100] <1>      mov     ecx, [MainProgCfg_mem_bsize] ; block size
6803      <1>
6804      <1>      ;call    deallocate_memory_block
6805      <1>      ;retn
6806 00009ECF E97FC0FFFF <1>      jmp     deallocate_memory_block
6807      <1>
6808      <1> loc_end_of_mainprog_cfg_line:
6809 00009ED4 C60700 <1>      mov     byte [edi], 0
6810      <1>
6811 00009ED7 8935[54770100] <1>      mov     [MainProgCfg_LineOffset], esi
6812      <1>
6813      <1>      ; 22/11/2017
6814 00009EDD BE[1A780100] <1>      mov     esi, CommandBuffer + 8
6815 00009EE2 29FE <1>      sub     esi, edi
6816 00009EE4 7606 <1>      jna     short loc_move_mainprog_cfg_command
6817 00009EE6 30C0 <1>      xor     al, al
6818      <1> loc_mainprog_cfg_clear_chrs:
6819 00009EE8 AA <1>      stosb
6820 00009EE9 4E <1>      dec     esi
6821 00009EEA 75FC <1>      jnz     short loc_mainprog_cfg_clear_chrs
6822      <1>
6823      <1> loc_move_mainprog_cfg_command:
6824 00009EEC BE[12780100] <1>      mov     esi, CommandBuffer
6825 00009EF1 89F7 <1>      mov     edi, esi
6826 00009EF3 31DB <1>      xor     ebx, ebx
6827      <1>      ;xor     ecx, ecx
6828 00009EF5 30C9 <1>      xor     cl, cl
6829      <1>
6830      <1> loc_move_mcfg_first_cmd_char:
6831 00009EF7 8A041E <1>      mov     al, [esi+ebx]
6832 00009EFA FEC3 <1>      inc     bl
6833 00009EFC 3C20 <1>      cmp     al, 20h
6834 00009EFE 7712 <1>      ja     short loc_move_mcfg_cmd_capitalizing
6835 00009F00 7237 <1>      jb     short loc_move_mcfg_cmd_arguments_ok
6836 00009F02 80FB4F <1>      cmp     bl, 79
6837 00009F05 72F0 <1>      jb     short loc_move_mcfg_first_cmd_char
6838 00009F07 EB30 <1>      jmp     short loc_move_mcfg_cmd_arguments_ok
6839      <1>
6840      <1> loc_move_mcfg_next_cmd_char:
6841 00009F09 8A041E <1>      mov     al, [esi+ebx]
6842 00009F0C FEC3 <1>      inc     bl
6843 00009F0E 3C20 <1>      cmp     al, 20h
6844 00009F10 7614 <1>      jna     short loc_move_mcfg_cmd_ok
6845      <1>
6846      <1> loc_move_mcfg_cmd_capitalizing:
6847 00009F12 3C61 <1>      cmp     al, 61h ; 'a'
6848 00009F14 7206 <1>      jb     short loc_move_mcfg_cmd_caps_ok
6849 00009F16 3C7A <1>      cmp     al, 7Ah ; 'z'
6850 00009F18 7702 <1>      ja     short loc_move_mcfg_cmd_caps_ok
6851 00009F1A 24DF <1>      and     al, 0DFh ; sub al, 'a'-'A'
6852      <1>
6853      <1> loc_move_mcfg_cmd_caps_ok:
6854 00009F1C AA <1>      stosb
6855 00009F1D FEC1 <1>      inc     cl
6856 00009F1F 80FB4F <1>      cmp     bl, 79
6857 00009F22 72E5 <1>      jb     short loc_move_mcfg_next_cmd_char
6858 00009F24 EB13 <1>      jmp     short loc_move_mcfg_cmd_arguments_ok
6859      <1>
6860      <1> loc_move_mcfg_cmd_ok:
6861 00009F26 30C0 <1>      xor     al, al ; 0
6862      <1>
6863      <1> loc_move_mcfg_cmd_arguments:
6864 00009F28 8807 <1>      mov     [edi], al
6865 00009F2A 47 <1>      inc     edi
6866 00009F2B 80FB4F <1>      cmp     bl, 79
6867 00009F2E 7309 <1>      jnb     short loc_move_mcfg_cmd_arguments_ok
6868 00009F30 8A041E <1>      mov     al, [esi+ebx]
6869 00009F33 FEC3 <1>      inc     bl
6870 00009F35 3C20 <1>      cmp     al, 20h
6871 00009F37 73EF <1>      jnb     short loc_move_mcfg_cmd_arguments
6872      <1>
6873      <1> loc_move_mcfg_cmd_arguments_ok:
6874 00009F39 C60700 <1>      mov     byte [edi], 0
6875      <1>
6876      <1> loc_mcfg_process_cmd_interpreter:
6877 00009F3C E828E1FFFF <1>      call    command_interpreter
6878      <1>
6879      <1> loc_mcfg_ci_return_addr:
6880 00009F41 A1[50770100] <1>      mov     eax, [MainProgCfg_FileSize]
6881 00009F46 89C2 <1>      mov     edx, eax
6882 00009F48 8B35[54770100] <1>      mov     esi, [MainProgCfg_LineOffset]
6883 00009F4E 01F2 <1>      add     edx, esi
6884      <1>      ;add     eax, [csftdf_sf_mem_addr]
6885      <1>      ; 19/12/2025
6886 00009F50 0305[7C820100] <1>      add     eax, [MainProgCfg_mem_addr]
6887 00009F56 29F0 <1>      sub     eax, esi
6888 00009F58 0F8730FFFFFF <1>      ja     loc_mcfg_process_next_line_check
6889      <1>
6890 00009F5E E861FFFFFF <1>      call    mcfg_deallocate_mem
6891      <1>
6892 00009F63 B94F000000 <1>      mov     ecx, 79 ; 80 ?
6893 00009F68 BF[12780100] <1>      mov     edi, CommandBuffer
6894 00009F6D 30C0 <1>      xor     al, al
6895 00009F6F F3AA <1>      rep     stosb
6896      <1>
6897      <1>      ; 06/05/2016
6898 00009F71 BE[07390100] <1>      mov     esi, nextline
6899 00009F76 E82FCEFFFF <1>      call    print_msg
6900 00009F7B E98FD8FFFF <1>      jmp     dos_prompt
6901      <1>
6902      <1>      ; 05/12/2025
6903      <1> ;mcfg_read_file_sectors:
6904      <1>      ; 14/04/2016
6905      <1>      ;cmp     byte [esi+LD_FATType], 0
6906      <1>      ;jna     short mcfg_read_fs_file_sectors
6907      <1>
6908      <1> mcfg_read_fat_file_sectors:
6909      <1>      ; return:
6910      <1>      ; CF = 0 & EDX > 0 -> END OF FILE
6911      <1>      ; CF = 0 & EDX = 0 -> not EOF
6912      <1>      ; CF = 1 -> read error (error code in AL)
6913      <1>
6914      <1> mcfg_read_fat_file_secs_0:
6915 00009F80 8B15[50770100] <1>      mov     edx, [MainProgCfg_FileSize]
6916      <1>      ;sub     edx, [csftdf_sf_rbytes]
6917      <1>      ; 19/12/2025
6918 00009F86 2B15[94820100] <1>      sub     edx, [MainProgCfg_rbytes]
6919      <1>      ;cmp     edx, [csftdf_r_size]
6920 00009F8C 3B15[8C820100] <1>      cmp     edx, [MainProgCfg_r_size]
6921 00009F92 7306 <1>      jnb     short mcfg_read_fat_file_secs_1

```

```

6922      <1>      ;mov     [csftdf_r_size], edx
6923      <1>      ; 19/12/2025
6924 00009F94 8915[8C820100] <1>      mov     [MainProgCfg_r_size], edx
6925      <1>
6926      <1> mcfg_read_fat_file_secs_1:
6927      <1>      ;mov     eax, [csftdf_r_size]
6928      <1>      ; 19/12/2025
6929 00009FA A1[8C820100] <1>      mov     eax, [MainProgCfg_r_size]
6930 00009F9F 29D2 <1>      sub     edx, edx
6931      <1>      ;movzx   ecx, word [esi+LD_BPB+BytesPerSec]
6932      <1>      ; 19/12/2025
6933 00009FA1 B900020000 <1>      mov     ecx, 512
6934 00009FA6 01C8 <1>      add     eax, ecx
6935 00009FA8 48 <1>      dec     eax
6936 00009FA9 F7F1 <1>      div     ecx
6937 00009FAB 89C1 <1>      mov     ecx, eax ; sector count
6938      <1>      ;mov     eax, [csftdf_sf_cluster]
6939      <1>      ; 19/12/2025
6940 00009FAD A1[84820100] <1>      mov     eax, [MainProgCfg_cluster]
6941      <1>
6942      <1>      ; EBX = memory block address (current)
6943      <1>
6944 00009FB2 E8B8210000 <1>      call    read_fat_file_sectors
6945 00009FB7 7230 <1>      jc      short mcfg_read_fat_file_secs_3
6946      <1>
6947      <1>      ; EBX = next memory address
6948      <1>
6949      <1>      ;mov     eax, [csftdf_sf_rbytes]
6950      <1>      ; 19/12/2025
6951 00009FB9 A1[94820100] <1>      mov     eax, [MainProgCfg_rbytes]
6952      <1>      ;add     eax, [csftdf_r_size]
6953 00009FBE 0305[8C820100] <1>      add     eax, [MainProgCfg_r_size]
6954 00009FC4 8B15[50770100] <1>      mov     edx, [MainProgCfg_FileSize]
6955 00009FCA 39D0 <1>      cmp     eax, edx
6956 00009FCC 731B <1>      jnb     short mcfg_read_fat_file_secs_3 ; edx > 0
6957      <1>      ;mov     [csftdf_sf_rbytes], eax
6958      <1>      ; 19/12/2025
6959 00009FCE A3[94820100] <1>      mov     [MainProgCfg_rbytes], eax
6960      <1>
6961 00009FD3 53 <1>      push    ebx ; *
6962      <1>      ; get next cluster (csftdf_r_size! bytes)
6963      <1>      ;mov     eax, [csftdf_sf_cluster]
6964      <1>      ; 19/12/2025
6965 00009FD4 A1[84820100] <1>      mov     eax, [MainProgCfg_cluster]
6966 00009FD9 E89B1F0000 <1>      call    get_next_cluster
6967 00009FDE 5B <1>      pop     ebx ; *
6968 00009FDF 7301 <1>      jnc     short mcfg_read_fat_file_secs_2
6969      <1>
6970      <1>      ;mov     eax, 17; Read error !
6971 00009FE1 C3 <1>      retn
6972      <1>
6973      <1> mcfg_read_fat_file_secs_2:
6974 00009FE2 29D2 <1>      sub     edx, edx ; 0
6975      <1>      ;mov     [csftdf_sf_cluster], eax ; next cluster
6976      <1>      ; 19/12/2025
6977 00009FE4 A3[84820100] <1>      mov     [MainProgCfg_cluster], eax
6978      <1>
6979      <1>      ; 25/07/2022 - TRDOS 386 Kernel v2.0.5
6980      <1>
6981      <1> mcfg_read_fat_file_secs_3:
6982      <1>      ; 19/12/2025
6983      <1>      ; ;retn
6984      <1>      ;
6985      <1> ;mcfg_read_fs_file_sectors:
6986      <1>      ; ;retn
6987      <1>      ;
6988      <1> ;loc_mcfg_load_fs_file:
6989 00009FE9 C3 <1>      retn
6990      <1>
6991      <1> load_and_execute_file:
6992      <1>      ; 02/12/2025
6993      <1>      ; 04/07/2025
6994      <1>      ; 03/07/2025 - TRDOS 386 Kernel v2.0.10
6995      <1>      ; 03/09/2024 - TRDOS 386 Kernel v2.0.9
6996      <1>      ; 30/08/2023 - TRDOS 386 Kernel v2.0.6
6997      <1>      ; 25/07/2022 - TRDOS 386 Kernel v2.0.5
6998      <1>      ; 04/01/2017
6999      <1>      ; 06/05/2016 - 07/05/2016 - 11/05/2016
7000      <1>      ; 23/04/2016 - 24/04/2016
7001      <1>      ; 22/04/2016 (TRDOS 386 = TRDOS v2.0)
7002      <1>      ; 05/11/2011
7003      <1>      ; (TRDOS v1, CMDINTR.ASM, 'cmp_cmd_run', 'cmp_cmd_external')
7004      <1>      ; ('loc_run_check_filename')
7005      <1>      ; 29/08/2011
7006      <1>      ; 10/09/2011
7007      <1>      ; INPUT->
7008      <1>      ; ESI = Path Name address (CommandBuffer address)
7009      <1>      ; OUTPUT ->
7010      <1>      ; none (error message will be shown if an error will occur)
7011      <1>      ;
7012      <1>      ; (EAX, EBX, ECX, EDX, ESI, EDI, EBP will be changed)
7013      <1>      ;
7014      <1> loc_run_check_filename:
7015 00009FEA 803E20 <1>      cmp     byte [esi], 20h
7016      <1>      ;jb     loc_cmd_failed
7017      <1>      ; 25/07/2022
7018 00009FED 7237 <1>      jb     short loc_run_ppn_failed
7019 00009FEF 7703 <1>      ja     short loc_run_check_filename_ok
7020 00009FF1 46 <1>      inc     esi
7021 00009FF2 EBF6 <1>      jmp     short loc_run_check_filename
7022      <1>
7023      <1> loc_run_check_filename_ok:
7024 00009FF4 C605[C3770100]00 <1>      mov     byte [CmdArgStart], 0 ; reset
7025 00009FFB 56 <1>      push    esi ; *
7026      <1> loc_run_get_first_arg_pos:
7027 00009FFC 46 <1>      inc     esi
7028 00009FFD 8A06 <1>      mov     al, [esi]
7029 00009FFF 3C20 <1>      cmp     al, 20h
7030 0000A001 77F9 <1>      ja     short loc_run_get_first_arg_pos
7031 0000A003 C60600 <1>      mov     byte [esi], 0
7032      <1> loc_run_get_external_arg_pos:
7033      <1>      ; 11/05/2016
7034 0000A006 46 <1>      inc     esi
7035 0000A007 8A06 <1>      mov     al, [esi]
7036 0000A009 3C20 <1>      cmp     al, 20h
7037 0000A00B 760C <1>      jna     short loc_run_parse_path_name
7038 0000A00D 89F0 <1>      mov     eax, esi
7039 0000A00F 2D[12780100] <1>      sub     eax, CommandBuffer
7040 0000A014 A2[C3770100] <1>      mov     byte [CmdArgStart], al
7041      <1> loc_run_parse_path_name:
7042      <1>      pop     esi ; *
7043 0000A01A BF[02800100] <1>      mov     edi, FindFile_Drv
7044 0000A01F E84A090000 <1>      call    parse_path_name
7045      <1>      ;jc     loc_cmd_failed

```

```

7046      <1>      ; 25/07/2022
7047 0000A024 7305      <1>      jnc      short loc_run_check_filename_exists
7048      <1>      loc_run_ppn_failed:
7049 0000A026 E919E4FFFF      <1>      jmp      loc_cmd_failed
7050      <1>
7051      <1>      loc_run_check_filename_exists:
7052 0000A02B BE[44800100]      <1>      mov     esi, FindFile_Name
7053 0000A030 803E20      <1>      cmp     byte [esi], 20h
7054      <1>      ;jna     loc_cmd_failed
7055      <1>      ; 25/07/2022
7056 0000A033 76F1      <1>      jna     short loc_run_ppn_failed
7057      <1>
7058      <1>      loc_run_check_exe_filename_ext:
7059 0000A035 E896020000      <1>      call    check_prg_filename_ext
7060      <1>      ;jc      loc_cmd_failed
7061      <1>      ; 25/07/2022
7062 0000A03A 72EA      <1>      jc      short loc_run_ppn_failed
7063      <1>      ; 03/07/2025
7064      <1>      ; if al = 0 -> ah = file name length
7065      <1>
7066      <1>      loc_run_check_exe_filename_ext_ok:
7067 0000A03C 66A3[1D830100]      <1>      mov     word [EXE_ID], ax ; 'P.' or 'P'+0
7068      <1>
7069      <1>      loc_run_drv:
7070 0000A042 C605[1C830100]00      <1>      mov     byte [Run_Manual_Path], 0
7071 0000A049 A1[5C770100]      <1>      mov     eax, [Current_Dir_FCluster]
7072 0000A04E A3[10830100]      <1>      mov     [Run_CDirFC], eax
7073      <1>
7074 0000A053 8A35[62770100]      <1>      mov     dh, [Current_Drv]
7075 0000A059 8835[BF7E0100]      <1>      mov     [RUN_CDRV], dh
7076      <1>
7077 0000A05F 8A15[02800100]      <1>      mov     dl, [FindFile_Drv]
7078 0000A065 38F2      <1>      cmp     dl, dh
7079 0000A067 7413      <1>      je      short loc_run_change_directory
7080      <1>
7081 0000A069 8005[1C830100]02      <1>      add     byte [Run_Manual_Path], 2
7082      <1>
7083 0000A070 E86BD6FFFF      <1>      call    change_current_drive
7084      <1>      ;jc      loc_run_cmd_failed
7085      <1>      ; 25/07/2022
7086 0000A075 7305      <1>      jnc     short loc_run_change_directory
7087 0000A077 E9F3E3FFFF      <1>      jmp     loc_run_cmd_failed
7088      <1>
7089      <1>      loc_run_change_directory:
7090 0000A07C 803D[03800100]20      <1>      cmp     byte [FindFile_Directory], 20h
7091 0000A083 7624      <1>      jna     short loc_run_find_executable_file
7092      <1>
7093 0000A085 FE05[1C830100]      <1>      inc     byte [Run_Manual_Path]
7094      <1>
7095 0000A08B FE05[7E2E0100]      <1>      inc     byte [Restore_CDIR]
7096      <1>
7097 0000A091 BE[03800100]      <1>      mov     esi, FindFile_Directory
7098 0000A096 30E4      <1>      xor     ah, ah ; CD_COMMAND sign -> 0
7099 0000A098 E83E030000      <1>      call    change_current_directory
7100      <1>      ;jc      loc_run_cmd_failed
7101      <1>      ; 25/07/2022
7102 0000A09D 7305      <1>      jnc     short loc_run_change_prompt_dir_string
7103 0000A09F E9CBE3FFFF      <1>      jmp     loc_run_cmd_failed
7104      <1>
7105      <1>      loc_run_change_prompt_dir_string:
7106 0000A0A4 E85B020000      <1>      call    change_prompt_dir_string
7107      <1>
7108      <1>      loc_run_find_executable_file:
7109 0000A0A9 66C705[18830100]00-      <1>      mov     word [Run_Auto_Path], 0
7109 0000A0B1 00      <1>
7110      <1>
7111      <1>      loc_run_find_executable_file_next:
7112 0000A0B2 BE[44800100]      <1>      mov     esi, FindFile_Name
7113      <1>      loc_run_find_program_file_next:
7114 0000A0B7 66B80018      <1>      mov     ax, 1800h ; Except volume label and dirs
7115 0000A0BB E8AAE8FFFF      <1>      call    find_first_file
7116      <1>      ; ESI = Directory Entry (FindFile_DirEntry) Location
7117      <1>      ; EDI = Directory Buffer Directory Entry Location
7118      <1>      ; EAX = File size
7119      <1>      ;jnc     loc_load_and_run_file
7120      <1>      ; 25/07/2022
7121 0000A0C0 7216      <1>      jc      short loc_run_program_file_not_found
7122      <1>
7123      <1>      ; 02/12/2025 (TRDOS 386 v2.0.10)
7124 0000A0C2 8B471C      <1>      mov     eax, [edi+DirEntry_FileSize]
7125 0000A0C5 21C0      <1>      and     eax, eax ; zero file size check?
7126 0000A0C7 750A      <1>      jnz     short loc_load_and_run_file_@
7127 0000A0C9 B814000000      <1>      mov     eax, ERR_FILE_SIZE
7128      <1>      ; 20, 'file size error !'
7129      <1>      ;mov     eax, ERR_ZERO_LENGTH
7130      <1>      ; 20, 'zero length !' error
7131 0000A0CE E99CE3FFFF      <1>      jmp     loc_run_cmd_failed
7132      <1>
7133      <1>      loc_load_and_run_file_@:
7134 0000A0D3 E957010000      <1>      jmp     loc_load_and_run_file
7135      <1>
7136      <1>      loc_run_program_file_not_found:
7137 0000A0D8 3C02      <1>      cmp     al, 2 ; file not found
7138      <1>      ;jne     loc_run_cmd_failed
7139      <1>      ; 25/07/2022
7140 0000A0DA 7405      <1>      je      short loc_run_progr_file_chk_prg_ext
7141 0000A0DC E98EE3FFFF      <1>      jmp     loc_run_cmd_failed
7142      <1>
7143      <1>      loc_run_progr_file_chk_prg_ext: ; 25/07/2022
7144 0000A0E1 66A1[1D830100]      <1>      mov     ax, word [EXE_ID]
7145      <1>
7146      <1>      ;cmp     ah, '.' ; File name has extension sign
7147      <1>      ;je      short loc_run_check_auto_path
7148      <1>      ;or      al, al
7149      <1>      ;jnz     short loc_run_check_auto_path
7150      <1>      ; 03/07/2025
7151 0000A0E7 663D502E      <1>      cmp     ax, 'P.'
7152 0000A0EB 7420      <1>      je      short loc_run_check_auto_path
7153      <1>
7154 0000A0ED 80FC08      <1>      cmp     ah, 8 ; count of file name chars
7155 0000A0F0 771B      <1>      ja      short loc_run_check_auto_path
7156      <1>
7157      <1>      loc_run_change_file_ext_to_prg:
7158 0000A0F2 0FB6DC      <1>      movzx   ebx, ah ; count of file name chars
7159 0000A0F5 BE[44800100]      <1>      mov     esi, FindFile_Name
7160 0000A0FA 01F3      <1>      add     ebx, esi
7161      <1>      ; 07/05/2016
7162 0000A0FC C7032E505247      <1>      mov     dword [ebx], '.PRG'
7163 0000A102 66C705[1D830100]50-      <1>      mov     word [EXE_ID], 'P.'
7163 0000A10A 2E      <1>
7164 0000A10B EBAA      <1>      jmp     short loc_run_find_program_file_next
7165      <1>
7166      <1>      loc_run_check_auto_path:
7167      <1>      ; NOTE: /// 07/05/2016 ///
```

```

7168      <1>      ; If the path is given, value of byte [Run_Manual_Path]
7169      <1>      ; will not be ZERO. If so, file searching by using
7170      <1>      ; Automatic Path (via 'PATH' environment variable)
7171      <1>      ; will not be applicable, because the program file
7172      <1>      ; is already/absolutely not found.
7173      <1>
7174      0000A10D A0[1C830100] <1>      mov     al, [Run_Manual_Path]
7175      0000A112 08C0 <1>      or      al, al
7176      <1>      ;jnz     loc_cmd_failed
7177      <1>      ; 25/07/2022
7178      0000A114 752B <1>      jnz     short loc_run_cap_failed
7179      <1>
7180      <1>      loc_run_check_auto_path_again:
7181      <1>      ;cmp     word [Run_Auto_Path], 0FFFFh
7182      <1>      ; 0FFFFh = Not a valid run path (in ENV block)
7183      <1>      ;jnb     loc_cmd_failed
7184      <1>      ; 25/07/2022
7185      <1>      ;jnb     short loc_run_cap_failed
7186      <1>      ; 03/07/2025
7187      0000A116 FF05[18830100] <1>      inc     dword [Run_Auto_Path] ; 0FFFFFFFh -> 0
7188      0000A11C 7423 <1>      jz      short loc_run_cap_failed ; -1 -> 0
7189      <1>
7190      <1>      ; 03/07/2025
7191      0000A11E BF[62780100] <1>      mov     edi, TextBuffer
7192      0000A123 FF0D[18830100] <1>      dec     dword [Run_Auto_Path] ; 0 -> init auto path
7193      0000A129 751D <1>      jnz     short loc_run_chk_filename_ext_next
7194      <1>
7195      <1>      ;xor     al, al
7196      0000A12B BE[532F0100] <1>      mov     esi, Cmd_Path ; 'PATH'
7197      <1>      ; 03/07/2025
7198      <1>      ;mov     edi, TextBuffer
7199      <1>      ; al = 0
7200      <1>      ; 03/09/2024 (bugfix)
7201      0000A130 66B90001 <1>      mov     cx, 256 ; TextBuffer (maximum) length
7202      0000A134 E86EF9FFFF <1>      call    get_environment_string
7203      0000A139 7315 <1>      jnc     short loc_run_chk_filename_ext_again
7204      <1>      ;mov     word [Run_Auto_Path], 0FFFFh ; invalid
7205      <1>      ; 03/07/2025
7206      0000A13B FF0D[18830100] <1>      dec     dword [Run_Auto_Path] ; -1 (invalid)
7207      <1>      loc_run_cap_failed: ; 25/07/2022
7208      <1>      ;jmp     loc_cmd_failed
7209      <1>      ; 30/08/2023
7210      0000A141 B001 <1>      mov     al, 1 ; (force jump to loc_cmd_failed at the end)
7211      0000A143 E927E3FFFF <1>      jmp     loc_run_cmd_failed ; (restore cdir if it is needed)
7212      <1>
7213      <1>      ; 03/07/2025
7214      <1>      loc_run_chk_filename_ext_next:
7215      0000A148 8B0D[14830100] <1>      mov     ecx, [Run_Path_Length]
7216      0000A14E EB09 <1>      jmp     short loc_run_chk_filename_ext_again_@
7217      <1>
7218      <1>      loc_run_chk_filename_ext_again:
7219      0000A150 89C1 <1>      mov     ecx, eax ; string length (with zero tail)
7220      0000A152 49 <1>      dec     ecx ; without zero tail
7221      <1>      ; 03/07/2025
7222      0000A153 890D[14830100] <1>      mov     [Run_Path_Length], ecx
7223      <1>      loc_run_chk_filename_ext_again_@:
7224      <1>      ; 04/07/2025
7225      <1>      ;mov     ax, [EXE_ID]
7226      <1>      ;cmp     ah, '.'
7227      <1>      ;je      short loc_run_chk_auto_path_pos
7228      <1>
7229      <1>      loc_run_change_file_ext_to_noext_again:
7230      <1>      ; 04/07/2025
7231      <1>      ;movzx   ebx, ah
7232      0000A159 BE[44800100] <1>      mov     esi, FindFile_Name
7233      <1>      ;add     ebx, esi
7234      0000A15E 31C0 <1>      xor     eax, eax
7235      <1>      loc_run_change_file_ext_to_noext_@:
7236      0000A160 46 <1>      inc     esi
7237      0000A161 FEC4 <1>      inc     ah
7238      <1>      ;cmp     al, '.'
7239      0000A163 803E2E <1>      cmp     byte [esi], '.'
7240      0000A166 75F8 <1>      jne     short loc_run_change_file_ext_to_noext_@
7241      0000A168 66A3[1D830100] <1>      mov     [EXE_ID], ax
7242      <1>      ; al = 0, ah = file name length
7243      <1>      ;sub     eax, eax
7244      <1>      ;mov     [ebx], eax ; 0 ; erase extension (.PRG)
7245      <1>      ;mov     [esi], eax ; 0
7246      0000A16E 8806 <1>      mov     [esi], al ; 0
7247      <1>
7248      <1>      loc_run_chk_auto_path_pos:
7249      <1>      ;movzx   eax, word [Run_Auto_Path]
7250      <1>      ;mov     ax, [Run_Auto_Path]
7251      <1>      ; 03/07/2025
7252      0000A170 A1[18830100] <1>      mov     eax, [Run_Auto_Path]
7253      0000A175 39C8 <1>      cmp     eax, ecx ; ecx = string length (except zero tail)
7254      <1>      ;jnb     loc_cmd_failed
7255      <1>      ; 25/07/2022
7256      0000A177 73C8 <1>      jnb     short loc_run_cap_failed
7257      <1>
7258      <1>      ; 03/07/2025
7259      0000A179 09C0 <1>      or      eax, eax
7260      <1>      ;or      ax, ax
7261      0000A17B 7502 <1>      jnz     short loc_run_auto_path_pos_move
7262      0000A17D B005 <1>      mov     al, 5
7263      <1>
7264      <1>      loc_run_auto_path_pos_move:
7265      0000A17F 89FE <1>      mov     esi, edi ; offset TextBuffer
7266      <1>      ; 04/07/2025
7267      0000A181 BF[E2780100] <1>      mov     edi, RunPathBuffer
7268      0000A186 01C6 <1>      add     esi, eax
7269      <1>
7270      <1>      loc_run_auto_path_pos_space_loop:
7271      0000A188 AC <1>      lodsb
7272      0000A189 3C20 <1>      cmp     al, 20h
7273      0000A18B 74FB <1>      je      short loc_run_auto_path_pos_space_loop
7274      <1>      ;jb     loc_cmd_failed
7275      <1>      ; 25/07/2022
7276      0000A18D 72B2 <1>      jb     short loc_run_cap_failed
7277      <1>
7278      <1>      loc_run_auto_path_pos_move_next_@:
7279      0000A18F AA <1>      stosb
7280      <1>      loc_run_auto_path_pos_move_next:
7281      0000A190 AC <1>      lodsb
7282      0000A191 3C3B <1>      cmp     al, ';'
7283      0000A193 741B <1>      je      short loc_run_auto_path_pos_move_last_byte
7284      0000A195 3C20 <1>      cmp     al, 20h
7285      0000A197 74F7 <1>      je      short loc_run_auto_path_pos_move_next
7286      <1>      ;jb     short loc_byte_ptr_end_of_path
7287      <1>      ;stosb
7288      <1>      ;jmp     short loc_run_auto_path_pos_move_next
7289      <1>      ; 03/09/2024
7290      0000A199 73F4 <1>      jnb     short loc_run_auto_path_pos_move_next_@
7291      <1>

```

```

7292                                     <1> loc_byte_ptr_end_of_path:
7293 0000A19B 66C705[18830100]FF- <1>     mov     word [Run_Auto_Path], 0FFFFh ; end of path
7293 0000A1A3 FF                    <1>
7294                                     <1>     ; 03/07/2025
7295 0000A1A4 C705[18830100]FFFF- <1>     mov     dword [Run_Auto_Path], 0FFFFFFFFh
7295 0000A1AC FFFF                  <1>
7296 0000A1AE EB0C                  <1>     jmp     short loc_run_auto_path_move_ok
7297                                     <1>
7298                                     <1> loc_run_auto_path_pos_move_last_byte:
7299 0000A1B0 89F0                  <1>     mov     eax, esi
7300 0000A1B2 2D[62780100]          <1>     sub     eax, TextBuffer
7301                                     <1>     ;mov     [Run_Auto_Path], ax ; next path position
7302                                     <1>     ; 03/07/2025
7303 0000A1B7 A3[18830100]          <1>     mov     [Run_Auto_Path], eax
7304                                     <1>
7305                                     <1> loc_run_auto_path_move_ok:
7306 0000A1BC 4F                    <1>     dec     edi
7307 0000A1BD B02F                  <1>     mov     al, '/'
7308 0000A1BF 3807                  <1>     cmp     [edi], al
7309 0000A1C1 7403                  <1>     je      short loc_run_auto_path_move_file_name
7310 0000A1C3 47                    <1>     inc     edi
7311 0000A1C4 8807                  <1>     mov     [edi], al
7312                                     <1>
7313                                     <1> loc_run_auto_path_move_file_name:
7314 0000A1C6 47                    <1>     inc     edi
7315 0000A1C7 BE[44800100]          <1>     mov     esi, FindFile_Name
7316                                     <1>
7317                                     <1> loc_run_auto_path_move_fn_loop:
7318 0000A1CC AC                    <1>     lodsb
7319 0000A1CD AA                    <1>     stosb
7320 0000A1CE 08C0                  <1>     or      al, al
7321 0000A1D0 75FA                  <1>     jnz     short loc_run_auto_path_move_fn_loop
7322                                     <1>
7323                                     <1>     ;mov     esi, TextBuffer
7324                                     <1>     ; 04/07/2025
7325 0000A1D2 BE[E2780100]          <1>     mov     esi, RunPathBuffer
7326 0000A1D7 BF[02800100]          <1>     mov     edi, FindFile_Drv
7327 0000A1DC E88D070000           <1>     call    parse_path_name
7328                                     <1>     ;jc      loc_run_cmd_failed
7329                                     <1>     ; 25/07/2022
7330                                     <1>     ;jnc     short loc_run_change_current_drive
7331                                     <1>     ;jmp     loc_run_cmd_failed
7332 0000A1E1 723D                  <1>     jc      short loc_run_path_failed
7333                                     <1>
7334                                     <1> loc_run_change_current_drive:
7335 0000A1E3 8A35[62770100]          <1>     mov     dh, [Current_Drv]
7336 0000A1E9 8A15[02800100]          <1>     mov     dl, [FindFile_Drv]
7337 0000A1EF 38F2                  <1>     cmp     dl, dh
7338 0000A1F1 7407                  <1>     je      short loc_run_change_directory_again
7339                                     <1>
7340 0000A1F3 E8E8D4FFFF           <1>     call    change_current_drive
7341                                     <1>     ;jc      loc_run_cmd_failed
7342                                     <1>     ; 25/07/2022
7343                                     <1>     ;jnc     short loc_run_change_directory_again
7344                                     <1>     ;jmp     loc_run_cmd_failed
7345 0000A1F8 7226                  <1>     jc      short loc_run_path_failed
7346                                     <1>
7347                                     <1> loc_run_change_directory_again:
7348 0000A1FA 803D[03800100]20        <1>     cmp     byte [FindFile_Directory], 20h
7349 0000A201 7627                  <1>     jna     short loc_load_executable_cdir_chk_again
7350                                     <1>
7351 0000A203 FE05[7E2E0100]          <1>     inc     byte [Restore_CDIR]
7352 0000A209 BE[03800100]          <1>     mov     esi, FindFile_Directory
7353 0000A20E 30E4                  <1>     xor     ah, ah ; CD_COMMAND sign -> 0
7354 0000A210 E8C6010000           <1>     call    change_current_directory
7355                                     <1>     ;jc      loc_run_cmd_failed
7356                                     <1>     ; 25/07/2022
7357 0000A215 730E                  <1>     jnc     short loc_run_chg_prompt_dir_str_again
7358                                     <1>
7359                                     <1>     ; 03/07/2025
7360                                     <1>     ; ignore 'directory not found' error
7361                                     <1>     ; if auto path in progress
7362 0000A217 803D[1C830100]00        <1>     cmp     byte [Run_Manual_Path], 0
7363 0000A21E 760A                  <1>     jna     short jmp_loc_run_find_executable_file_next
7364                                     <1>
7365                                     <1> loc_run_path_failed: ; 25/07/2022
7366 0000A220 E94AE2FFFF           <1>     jmp     loc_run_cmd_failed
7367                                     <1>
7368                                     <1> loc_run_chg_prompt_dir_str_again:
7369 0000A225 E8DA000000           <1>     call    change_prompt_dir_string
7370                                     <1>
7371                                     <1> loc_load_executable_cdir_chk_again:
7372                                     <1>     ; 04/07/2025
7373                                     <1>     %if 0
7374                                     <1>     ; 03/07/2025
7375                                     <1>     mov     al, [Current_Drv]
7376                                     <1>     cmp     al, [RUN_CDRV]
7377                                     <1>     jne     short jmp_loc_run_find_executable_file_next
7378                                     <1>
7379                                     <1>     mov     eax, [Current_Dir_FCluster]
7380                                     <1>     cmp     eax, [Run_CDirFC]
7381                                     <1>     ;jne     loc_run_find_executable_file_next
7382                                     <1>     ; 30/08/2023
7383                                     <1>     je      short jmp_loc_run_check_auto_path_again
7384                                     <1>
7385                                     <1> jmp_loc_run_find_executable_file_next: ; 03/07/2025
7386                                     <1>     jmp     loc_run_find_executable_file_next
7387                                     <1> jmp_loc_run_check_auto_path_again:
7388                                     <1>     xor     al, al ; 0
7389                                     <1>     jmp     loc_run_check_auto_path_again
7390                                     <1> %else
7391                                     <1>     ; 04/07/2025
7392                                     <1> jmp_loc_run_find_executable_file_next:
7393 0000A22A E983FEFFFF           <1>     jmp     loc_run_find_executable_file_next
7394                                     <1> %endif
7395                                     <1>
7396                                     <1> loc_load_and_run_file:
7397                                     <1>     ; 25/07/2022 - TRDOS 386 kernel v2.0.5
7398                                     <1>     ; 13/11/2017
7399                                     <1>     ; 04/01/2017
7400                                     <1>     ; 23/04/2016
7401 0000A22F BE[44800100]          <1>     mov     esi, FindFile_Name
7402 0000A234 BF[62780100]          <1>     mov     edi, TextBuffer
7403                                     <1>
7404                                     <1>     ; 24/04/2016
7405 0000A239 31D2                  <1>     xor     edx, edx
7406                                     <1>     ;mov     word [argc], dx ; 0
7407                                     <1>     ; 25/07/2022
7408 0000A23B 8915[508F0100]          <1>     mov     dword [argc], edx
7409 0000A241 8915[808E0100]          <1>     mov     dword [u.nread], edx ; 0
7410                                     <1>
7411                                     <1> loc_load_and_run_file_1:
7412 0000A247 AC                    <1>     lodsb
7413 0000A248 AA                    <1>     stosb

```

```

7414 0000A249 FF05[808E0100] <1> inc dword [u.nread]
7415 0000A24F 20C0 <1> and al, al
7416 0000A251 75F4 <1> jnz short loc_load_and_run_file_1
7417 <1>
7418 0000A253 A0[C3770100] <1> mov al, [CmdArgStart]
7419 0000A258 20C0 <1> and al, al
7420 0000A25A 7442 <1> jz short loc_load_and_run_file_7
7421 <1>
7422 0000A25C 0FB6F0 <1> movzx esi, al ; 11/05/2016
7423 <1> ;mov ecx, 80
7424 <1> ; 25/07/2022
7425 0000A25F 31C9 <1> xor ecx, ecx
7426 0000A261 B150 <1> mov cl, 80
7427 <1> ;sub ecx, esi
7428 0000A263 28C1 <1> sub cl, al
7429 0000A265 81C6[12780100] <1> add esi, CommandBuffer
7430 <1>
7431 <1> ;inc word [argc] ; 11/05/2016
7432 <1> ; 25/07/2022
7433 0000A26B FF05[508F0100] <1> inc dword [argc]
7434 <1>
7435 <1> loc_load_and_run_file_2:
7436 0000A271 AC <1> lodsb
7437 0000A272 3C20 <1> cmp al, 20h
7438 0000A274 7716 <1> ja short loc_load_and_run_file_5
7439 0000A276 721D <1> jb short loc_load_and_run_file_6
7440 <1>
7441 <1> loc_load_and_run_file_3:
7442 0000A278 803E20 <1> cmp byte [esi], 20h
7443 0000A27B 7707 <1> ja short loc_load_and_run_file_4
7444 0000A27D 7216 <1> jb short loc_load_and_run_file_6
7445 0000A27F 46 <1> inc esi
7446 0000A280 E2F6 <1> loop loc_load_and_run_file_3
7447 0000A282 EB11 <1> jmp short loc_load_and_run_file_6
7448 <1>
7449 <1> loc_load_and_run_file_4:
7450 0000A284 28C0 <1> sub al, al ; 0
7451 <1> ;inc word [argc]
7452 <1> ; 25/07/2022
7453 0000A286 FF05[508F0100] <1> inc dword [argc]
7454 <1> loc_load_and_run_file_5:
7455 0000A28C AA <1> stosb
7456 0000A28D FF05[808E0100] <1> inc dword [u.nread]
7457 0000A293 E2DC <1> loop loc_load_and_run_file_2
7458 <1>
7459 <1> loc_load_and_run_file_6:
7460 0000A295 30C0 <1> xor al, al ; 0
7461 0000A297 AA <1> stosb
7462 0000A298 FF05[808E0100] <1> inc dword [u.nread]
7463 <1> loc_load_and_run_file_7:
7464 0000A29E 8807 <1> mov [edi], al ; 0
7465 <1> ;inc word [argc] ; 24/04/2016
7466 <1> ; 25/07/2022
7467 0000A2A0 FF05[508F0100] <1> inc dword [argc]
7468 0000A2A6 FF05[808E0100] <1> inc dword [u.nread] ; 24/04/2016
7469 0000A2AC BE[62780100] <1> mov esi, TextBuffer
7470 0000A2B1 8B15[70800100] <1> mov edx, [FindFile_DirEntry+DirEntry_FileSize]
7471 0000A2B7 66A1[68800100] <1> mov ax, [FindFile_DirEntry+DirEntry_FstClusHI]
7472 0000A2BD C1E010 <1> shl eax, 16 ; 13/11/2017
7473 0000A2C0 66A1[6E800100] <1> mov ax, [FindFile_DirEntry+DirEntry_FstClusLO]
7474 <1> ; EAX = First Cluster number
7475 <1> ; EDX = File Size
7476 <1> ; ESI = Argument list address
7477 <1> ; [argc] = argument count
7478 <1> ; [u.nread] = argument list length
7479 0000A2C6 E8BA620000 <1> call load_and_run_file ; trdosk6.s
7480 <1> ;jc loc_run_cmd_failed ; 04/01/2017
7481 <1> loc_load_and_run_file_8: ; 06/05/2016
7482 0000A2CB E968EAFFFF <1> jmp loc_file_rw_restore_retn
7483 <1>
7484 <1> check_prg_filename_ext:
7485 <1> ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)
7486 <1> ; 10/09/2011
7487 <1> ; (TRDOS v1, CMDINTR.ASM, 'proc_check_exe_filename_ext')
7488 <1> ; 14/11/2009
7489 <1> ; INPUT ->
7490 <1> ; ESI = Dot File Name
7491 <1> ; OUTPUT ->
7492 <1> ; cf = 0 -> EXE_ID in AL
7493 <1> ; ESI = Last char + 1 position
7494 <1> ; cf = 1 -> Invalid executable file name
7495 <1> ; or no file name extension if AH<=8
7496 <1> ; AL = Last file name char
7497 <1> ; cf = 0 -> AL='P' (PRG), AL=0 (no extension)
7498 <1> ;
7499 <1> ; (Modified registers: EAX, ESI)
7500 <1>
7501 0000A2D0 30E4 <1> xor ah, ah
7502 <1> loc_run_check_filename_ext:
7503 0000A2D2 AC <1> lodsb
7504 0000A2D3 3C21 <1> cmp al, 21h
7505 0000A2D5 7229 <1> jb short loc_check_exe_fn_retn
7506 0000A2D7 FEC4 <1> inc ah
7507 0000A2D9 3C2E <1> cmp al, '.'
7508 0000A2DB 75F5 <1> jne short loc_run_check_filename_ext
7509 <1>
7510 <1> loc_run_check_filename_ext_dot:
7511 0000A2DD 80FC02 <1> cmp ah, 2 ; ??? is not valid
7512 0000A2E0 88C4 <1> mov ah, al ; '.'
7513 0000A2E2 7219 <1> jb short loc_check_prg_fn_retn
7514 <1>
7515 <1> loc_run_check_filename_ext_dot_ok:
7516 0000A2E4 AC <1> lodsb
7517 0000A2E5 24DF <1> and al, 0DFh
7518 <1>
7519 <1> loc_run_check_filename_ext_prg:
7520 0000A2E7 3C50 <1> cmp al, 'P'
7521 0000A2E9 7212 <1> jb short loc_check_prg_fn_retn
7522 0000A2EB 7711 <1> ja short loc_check_prg_fn_stc
7523 0000A2ED AC <1> lodsb
7524 0000A2EE 24DF <1> and al, 0DFh
7525 0000A2F0 3C52 <1> cmp al, 'R'
7526 0000A2F2 750A <1> jne short loc_check_prg_fn_stc
7527 0000A2F4 AC <1> lodsb
7528 0000A2F5 24DF <1> and al, 0DFh
7529 0000A2F7 3C47 <1> cmp al, 'G'
7530 0000A2F9 7503 <1> jne short loc_check_prg_fn_stc
7531 <1>
7532 0000A2FB B050 <1> mov al, 'P'
7533 <1> loc_check_prg_fn_retn:
7534 0000A2FD C3 <1> retn
7535 <1>
7536 <1> ; 25/07/2022 - TRDOS 386 Kernel v2.0.5
7537 <1>

```

```

7538      <1> loc_check_prg_fn_stc:
7539 0000A2FE F9      <1>      stc
7540 0000A2FF C3      <1>      retn
7541      <1>
7542      <1> loc_check_exe_fn_retn:
7543 0000A300 28C0     <1>      sub    al, al ; 0
7544      <1>      ;retn
7545      <1>
7546      <1> find_and_list_files:
7547      <1>      ;retn
7548      <1> set_exec_arguments:
7549 0000A302 C3      <1>      retn
7550      <1>
7551      <1> ; 19/12/2025
7552      <1> ;delete_fs_directory:
7553      <1> ;    xor    eax, eax
7554 0000A303 C3      <1>      retn
3435      <1> %include 'trdosk4.s' ; 24/01/2016
      <1> ; *****
      <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.10) - Directory Functions : trdosk4.s
      <1> ; -----
      <1> ; Last Update: 19/12/2025 (Previous: 03/09/2024, v2.0.9)
      <1> ; -----
      <1> ; Beginning: 24/01/2016
      <1> ; -----
      <1> ; Assembler: NASM version 2.15 (trdos386.s)
      <1> ; -----
      <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
      <1> ; DIR.ASM (09/10/2011)
      <1> ; *****
      <1> ;
      <1> ; DIR.ASM [ TRDOS KERNEL - COMMAND EXECUTER SECTION - DIRECTORY FUNCTIONS ]
      <1> ; (c) 2004-2010 Erdogan TAN [ 17/01/2004 ] Last Update: 09/10/2011
      <1> ; FILE.ASM [ FILE FUNCTIONS ] Last Update: 09/10/2011
      <1> ;
      <1> change_prompt_dir_string:
      <1> ; 05/10/2016
      <1> ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)
      <1> ; 27/03/2011
      <1> ; 09/10/2009
      <1> ; INPUT/OUTPUT => none
      <1> ; this procedure changes current directory string/text
      <1> ; 2005
      <1>
      <1> mov     esi, PATH_Array
      <1> change_prompt_dir_str: ; 05/10/2016 (call from 'set_working_path')
      <1> mov     edi, Current_Directory
      <1> mov     ah, [Current_Dir_Level]
      <1> call    set_current_directory_string
      <1> mov     [Current_Dir_StrLen], cl
      <1>
      <1> retn
      <1>
      <1> set_current_directory_string:
      <1> ; 11/08/2022 (TRDOS 386 kernel v2.0.5)
      <1> ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)
      <1> ; 27/03/2011
      <1> ; 09/10/2009
      <1> ; INPUT:
      <1> ;     ESI = Path Array Address
      <1> ;     EDI = Current Directory String Buffer
      <1> ;     AH = Current Directory Level
      <1> ; OUTPUT => EAX, EBX, ESI will be changed
      <1> ;     EDI will be same with input
      <1> ;     ECX = Current Directory String Length
      <1>
      <1> push    edi
      <1> cmp     ah, 0
      <1> jna     short pass_write_path
      <1> add     esi, 16
      <1> mov     ebx, esi
      <1> loc_write_path:
      <1> ;mov     ecx, 8
      <1> ; 11/08/2022
      <1> sub     ecx, ecx
      <1> mov     cl, 8
      <1> path_write_dirname1:
      <1> lodsb
      <1> cmp     al, 20h
      <1> jna     short pass_write_dirname1
      <1> stosb
      <1> cmp     edi, End_Of_Current_Dir_Str
      <1> jnb     short pass_write_path
      <1> loop    path_write_dirname1
      <1> cmp     byte [esi], 20h
      <1> jna     short pass_write_dirname2
      <1> jmp     short loc_put_dot_cont_ext
      <1> pass_write_dirname1:
      <1> mov     esi, ebx
      <1> add     esi, 8
      <1> cmp     byte [esi], 20h
      <1> jna     short pass_write_dirname2
      <1> loc_put_dot_cont_ext:
      <1> mov     byte [edi], "."
      <1> ;mov     ecx, 3
      <1> mov     cl, 3
      <1> loc_check_dir_name_ext:
      <1> lodsb
      <1> inc     edi
      <1> cmp     al, 20h
      <1> jna     short pass_write_dirname2
      <1> mov     [edi], al
      <1> cmp     edi, End_Of_Current_Dir_Str
      <1> jnb     short pass_write_path
      <1> loop    loc_check_dir_name_ext
      <1> inc     edi
      <1> pass_write_dirname2:
      <1> dec     ah
      <1> jz      short pass_write_path
      <1> add     ebx, 16
      <1> mov     esi, ebx
      <1> mov     byte [edi], "/"
      <1> inc     edi
      <1> jmp     short loc_write_path
      <1> pass_write_path:
      <1> mov     byte [edi], 0
      <1> inc     edi
      <1> mov     ecx, edi
      <1> pop     edi
      <1> sub     ecx, edi
      <1> ; ECX = Current Directory String Length
      <1>
      <1> retn
      <1>
      <1> get_current_directory:

```



```

107      <1>      ; 29/08/2023 (TRDOS 386 kernel v2.0.6)
108      <1>      ; 11/08/2022
109      <1>      ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
110      <1>      ; 15/10/2016
111      <1>      ; 14/02/2016
112      <1>      ; 24/01/2016 (TRDOS 386 = TRDOS v2.0)
113      <1>      ; 27/03/2011
114      <1>      ;
115      <1>      ; INPUT-> ESI = Current Directory Buffer
116      <1>      ;         DL = TRDOS Logical Dos Drive Number + 1
117      <1>      ;         (0 = Default/Current Drive)
118      <1>      ;
119      <1>      ; Note: Required dir buffer length may be <= 92 bytes
120      <1>      ;         for TRDOS (7*12 name chars + 7 slash + 0)
121      <1>      ; OUTPUT -> ESI = Current Directory Buffer
122      <1>      ;         EAX, EBX, ECX, EDX, EDI will be changed
123      <1>      ;         CX/CL = Current Directory String Length
124      <1>      ;         DL = Drive Number (0 based)
125      <1>      ;         (If input is 0, output is current drv number)
126      <1>      ;         DH = same with input
127      <1>      ;         cf = 0 -> AL = 0
128      <1>      ;         cf = 1 -> error code in AL
129      <1>      ;
130      <1>      loc_get_current_drive_0:
131      <1>      ; 29/08/2023
132      0000A381 29C0      <1>      sub     eax, eax ; 0
133      <1>      ; cmp     dl, 0
134      0000A383 38C2      <1>      cmp     dl, al
135      0000A385 7724      <1>      ja      short loc_get_current_drive_1
136      0000A387 8A15[62770100] <1>      mov     dl, [Current_Drv]
137      <1>      ; 29/08/2023
138      <1>      ; jmp     short loc_get_current_drive_2
139      <1>      ;
140      <1>      loc_get_current_drive_2:
141      <1>      ; 29/08/2023
142      <1>      ; xor     eax, eax
143      <1>      ; eax = 0
144      0000A38D 88D4      <1>      mov     ah, dl
145      0000A38F 56          <1>      push    esi ; (*)
146      0000A390 BE00010900 <1>      mov     esi, Logical_DOSDisks
147      0000A395 01C6      <1>      add     esi, eax
148      0000A397 8A06      <1>      mov     al, [esi+LD_Name]
149      0000A399 3C41      <1>      cmp     al, 'A'
150      0000A39B 721C      <1>      jnb     short loc_get_current_drive_not_ready_retn
151      <1>      ;
152      <1>      ; 28/07/2022
153      0000A39D 31C9      <1>      xor     ecx, ecx
154      <1>      ;
155      0000A39F 8A667F      <1>      mov     ah, [esi+LD_CDirLevel]
156      0000A3A2 08E4      <1>      or      ah, ah
157      0000A3A4 7519      <1>      jnz     short loc_get_current_drive_3
158      <1>      ;
159      <1>      ; xor     ah, ah ; mov ah, 0
160      <1>      ;
161      <1>      ; 11/08/2022 - BugFix (*)
162      0000A3A6 5E          <1>      pop     esi ; (*) Current Directory Buffer address
163      <1>      ;
164      0000A3A7 8826      <1>      mov     [esi], ah
165      <1>      ; 28/07/2022
166      <1>      ; xor     ecx, ecx
167      0000A3A9 EB2D      <1>      jmp     short loc_get_current_drive_4
168      <1>      ;
169      <1>      ; 29/08/2023
170      <1>      loc_get_current_drive_1:
171      0000A3AB FECA      <1>      dec     dl
172      0000A3AD 3A15[7D2E0100] <1>      cmp     dl, [Last_DOS_DiskNo]
173      0000A3B3 76D8      <1>      jna     short loc_get_current_drive_2
174      <1>      ; mov     eax, 0Fh ; Invalid drive (Drive not ready!)
175      <1>      ; cmc     ; stc
176      <1>      ; 28/07/2022
177      <1>      ; sub     eax, eax ; 29/08/2023
178      <1>      ; eax = 0
179      0000A3B5 B00F      <1>      mov     al, 0Fh
180      0000A3B7 F9          <1>      stc
181      0000A3B8 C3          <1>      retn
182      <1>      ;
183      <1>      loc_get_current_drive_not_ready_retn:
184      0000A3B9 5E          <1>      pop     esi
185      <1>      ; mov     eax, 15
186      0000A3BA 66B80F00 <1>      mov     ax, 15 ; Drive not ready
187      0000A3BE C3          <1>      retn
188      <1>      ;
189      <1>      loc_get_current_drive_3:
190      0000A3BF BFC07E0100] <1>      mov     edi, PATH_Array
191      0000A3C4 57          <1>      push    edi
192      0000A3C5 81C680000000 <1>      add     esi, LD_CurrentDirectory
193      <1>      ; mov     ecx, 32
194      <1>      ; 28/07/2022
195      0000A3CB B120      <1>      mov     cl, 32
196      0000A3CD F3A5      <1>      rep     movsd
197      0000A3CF 5E          <1>      pop     esi ; Path Array Address
198      0000A3D0 5F          <1>      pop     edi ; pushed esi (current dir buffer offset)
199      <1>      ;
200      0000A3D1 E84AFFFFFF <1>      call    set_current_directory_string
201      0000A3D6 89FE      <1>      mov     esi, edi
202      <1>      ;
203      <1>      loc_get_current_drive_4:
204      0000A3D8 30C0      <1>      xor     al, al
205      0000A3DA C3          <1>      retn
206      <1>      ;
207      <1>      change_current_directory:
208      <1>      ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
209      <1>      ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
210      <1>      ; 02/03/2021 (TRDOS 386 v2.0.3) ((BugFix))
211      <1>      ; 19/02/2016
212      <1>      ; 11/02/2016
213      <1>      ; 10/02/2016
214      <1>      ; 08/02/2016
215      <1>      ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
216      <1>      ; 18/09/2011 (DIR.ASM, 09/10/2011)
217      <1>      ; 04/10/2009
218      <1>      ; 2005
219      <1>      ; INPUT ->
220      <1>      ;     ESI = Directory string
221      <1>      ;     ah = CD command (CDh = save current dir string)
222      <1>      ; OUTPUT ->
223      <1>      ;     EDI = DOS Drive Description Table
224      <1>      ;     cf = 1 -> error
225      <1>      ;     EAX = Error code
226      <1>      ;     cf = 0 -> successful
227      <1>      ;     ESI = PATH_Array
228      <1>      ;     EAX = Current Directory First Cluster
229      <1>      ;
230      <1>      ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)

```

```

231                                     <1>
232 0000A3DB 8825[4E7F0100]          <1>      mov     [CD_COMMAND], ah
233 0000A3E1 803E2F                  <1>      cmp     byte [esi], '/'
234 0000A3E4 7505                    <1>      jne     short loc_ccd_cdir_level
235 0000A3E6 46                      <1>      inc     esi
236 0000A3E7 30C0                    <1>      xor     al, al
237 0000A3E9 EB05                    <1>      jmp     short loc_ccd_parse_path_name
238                                     <1> loc_ccd_cdir_level:
239 0000A3EB A0[60770100]              <1>      mov     al, [Current_Dir_Level]
240                                     <1> loc_ccd_parse_path_name:
241 0000A3F0 88C4                    <1>      mov     ah, al
242 0000A3F2 BF[C07E0100]              <1>      mov     edi, PATH_Array
243                                     <1>
244                                     <1> ; Reset directory levels > cdir level
245                                     <1> ; is this required !?
246                                     <1> ;
247                                     <1> ; Relations:
248                                     <1> ; MAINPROG.ASM (pass_ccdrv_reset_cdir_FAT_fcluster)
249                                     <1> ; proc_parse_dir_name,
250                                     <1> ; proc_change_current_directory (this procedure)
251                                     <1> ; proc_change_prompt_dir_string
252                                     <1>
253 0000A3F7 0FB6C8                  <1>      movzx   ecx, al
254 0000A3FA FEC1                    <1>      inc     cl
255 0000A3FC C0E104                  <1>      shl     cl, 4
256 0000A3FF 01CF                    <1>      add     edi, ecx
257 0000A401 B107                    <1>      mov     cl, 7
258 0000A403 28C1                    <1>      sub     cl, al
259 0000A405 C0E102                  <1>      shl     cl, 2
260 0000A408 89C3                    <1>      mov     ebx, eax
261 0000A40A 31C0                    <1>      xor     eax, eax ; 0
262 0000A40C F3AB                    <1>      rep     stosd
263 0000A40E 89D8                    <1>      mov     eax, ebx
264                                     <1>
265 0000A410 BF[C07E0100]              <1>      mov     edi, PATH_Array
266                                     <1>
267 0000A415 803E20                  <1>      cmp     byte [esi], 20h
268 0000A418 F5                     <1>      cmc
269 0000A419 7305                    <1>      jnc     short pass_ccd_parse_dir_name
270                                     <1>
271                                     <1> ; ESI = Path name
272                                     <1> ; AL = CCD_Level
273 0000A41B E83B010000              <1>      call    parse_dir_name
274                                     <1> ; AL = CCD_Level
275                                     <1> ; AH = Last_Dir_Level
276                                     <1> ; (EDI = PATH_Array)
277                                     <1>
278                                     <1> pass_ccd_parse_dir_name:
279 0000A420 9C                      <1>      pushf
280                                     <1>
281                                     <1> ;mov     [CCD_Level], al
282                                     <1> ;mov     [Last_Dir_Level], ah
283 0000A421 66A3[447F0100]          <1>      mov     [CCD_Level], ax
284                                     <1>
285 0000A427 31DB                    <1>      xor     ebx, ebx
286 0000A429 8A3D[62770100]          <1>      mov     bh, [Current_Drv]
287 0000A42F BE00010900              <1>      mov     esi, Logical_DOSDisks
288 0000A434 01DE                    <1>      add     esi, ebx
289                                     <1>
290 0000A436 9D                      <1>      popf
291 0000A437 720A                    <1>      jc      short loc_ccd_bad_path_name_retn
292                                     <1>
293 0000A439 8935[407F0100]          <1>      mov     [CCD_DriveDT], esi
294                                     <1>
295 0000A43F 3C07                    <1>      cmp     al, 7
296 0000A441 7208                    <1>      jb      short loc_ccd_load_child_dir
297                                     <1>
298                                     <1> loc_ccd_bad_path_name_retn:
299 0000A443 87FE                    <1>      xchg     esi, edi
300                                     <1> ;mov     eax, 19 ; Bad directory/path name
301                                     <1> ; 28/07/2022
302 0000A445 29C0                    <1>      sub     eax, eax
303 0000A447 B013                    <1>      mov     al, 19
304 0000A449 F9                     <1>      stc
305                                     <1> loc_ccd_retn_p:
306 0000A44A C3                      <1>      retn
307                                     <1>
308                                     <1> loc_ccd_load_child_dir:
309                                     <1> ; AL = CCD_Level
310 0000A44B 08C0                    <1>      or      al, al
311 0000A44D 744C                    <1>      jz      short loc_ccd_load_root_dir
312                                     <1>
313                                     <1> ;mov     cx, ax
314                                     <1> ; 28/07/2022
315 0000A44F 89C1                    <1>      mov     ecx, eax
316 0000A451 C0E004                  <1>      shl     al, 4
317 0000A454 0FB6F0                  <1>      movzx   esi, al
318 0000A457 01FE                    <1>      add     esi, edi ; offset PATH_Array
319                                     <1>
320 0000A459 8B460C                  <1>      mov     eax, [esi+12]
321 0000A45C 38E9                    <1>      cmp     cl, ch
322                                     <1> ;je      loc_ccd_load_sub_directory
323                                     <1> ; 28/07/2022
324 0000A45E 7505                    <1>      jne     short loc_ccd_1
325 0000A460 E992000000              <1>      jmp     loc_ccd_load_sub_directory
326                                     <1>
327                                     <1> loc_ccd_1: ; 28/07/2022
328 0000A465 A3[5C770100]              <1>      mov     [Current_Dir_FCluster], eax
329                                     <1>
330                                     <1> loc_ccd_load_child_dir_next:
331 0000A46A 83C610                  <1>      add     esi, 16 ; DOS DirEntry Format FileName Address
332                                     <1>
333                                     <1> ; Directory attribute : 10h
334 0000A46D B010                    <1>      mov     al, 00010000b ; 10h (Attrib AND mask)
335                                     <1> ;mov     ah, 11001000b ; C8h
336                                     <1> ; Volume name attribute: 8h
337 0000A46F B408                    <1>      mov     ah, 00001000b ; 08h (Attrib NAND, AND --> zero mask)
338                                     <1>
339                                     <1> ;xor     cx, cx
340 0000A471 31C9                    <1>      xor     ecx, ecx ; 02/03/2021
341 0000A473 E880010000              <1>      call    locate_current_dir_file
342 0000A478 733C                    <1>      jnc     short loc_ccd_set_dir_cluster_ptr
343                                     <1>
344                                     <1> ; 19/02/2016
345                                     <1> ;mov     edi, [CCD_DriveDT]
346 0000A47A 8A25[447F0100]          <1>      mov     ah, [CCD_Level]
347 0000A480 803D[4E7F0100]CD        <1>      cmp     byte [CD_COMMAND], 0Cdh ; 'CD' command or another
348 0000A487 7509                    <1>      jne     short loc_ccd_load_child_dir_err
349                                     <1> ; It is better to save recent successful part
350                                     <1> ; of the (requested) path as current directory.
351                                     <1> ; (Otherwise the path would be reset to back
352                                     <1> ; on the next 'CD' command.)
353 0000A489 88E1                    <1>      mov     cl, ah
354 0000A48B 50                      <1>      push    eax

```

```

355 0000A48C E8AE000000    <1>    call    loc_ccd_save_current_dir
356 0000A491 58           <1>    pop     eax
357                   <1> loc_ccd_load_child_dir_err:
358 0000A492 3C03         <1>    cmp     al, 3    ; AL = 2 => File not found error
359 0000A494 7202         <1>    jnb    short loc_ccd_path_not_found_retn
360 0000A496 F9           <1>    stc
361 0000A497 C3           <1>    retn
362                   <1>
363                   <1> loc_ccd_path_not_found_retn:
364 0000A498 B003         <1>    mov     al, 3    ; Path not found
365 0000A49A C3           <1>    retn
366                   <1>
367                   <1> loc_ccd_load_root_dir:    ; 19/12/2025
368                   <1> loc_ccd_load_FAT_root_dir:
369 0000A49B 803D[61770100]02 <1>    cmp     byte [Current_FATType], 2
370 0000A4A2 774B         <1>    ja     short loc_ccd_load_FAT32_root_dir
371                   <1>
372                   <1> ;mov     esi, [CCD_DriveDT]
373                   <1> ;push    esi
374 0000A4A4 E80E1C0000    <1>    call    load_FAT_root_directory
375                   <1> ;pop     edi ; Dos Drv Description Table
376                   <1>
377 0000A4A9 89F7         <1>    mov     edi, esi
378 0000A4AB BE[C07E0100]    <1>    mov     esi, PATH_Array
379 0000A4B0 7298         <1>    jc     short loc_ccd_retn_p
380                   <1>
381 0000A4B2 31C0         <1>    xor     eax, eax
382 0000A4B4 EB58         <1>    jmp     short loc_ccd_set_cdfc
383                   <1>
384                   <1> ; 19/12/2025
385                   <1> %if 0
386                   <1> loc_ccd_load_root_dir:
387                   <1>    cmp     byte [Current_FATType], 1
388                   <1>    jnb    short loc_ccd_load_FAT_root_dir
389                   <1>
390                   <1> loc_ccd_load_FS_root_dir:
391                   <1>    call    load_FS_root_directory
392                   <1>    jmp     short pass_ccd_load_FAT_sub_directory
393                   <1>
394                   <1> loc_ccd_load_FS_sub_directory_next:
395                   <1>    call    load_FS_sub_directory
396                   <1>    jmp     short pass_ccd_set_dir_cluster_ptr
397                   <1> %endif
398                   <1>
399                   <1> loc_ccd_set_dir_cluster_ptr:
400                   <1> ; EDI = Directory Entry
401 0000A4B6 668B4714    <1>    mov     ax, [edi+20] ; First Cluster High word
402 0000A4BA C1E010         <1>    shl     eax, 16
403 0000A4BD 668B471A    <1>    mov     ax, [edi+26] ; First Cluster Low word
404                   <1>
405 0000A4C1 8B35[407F0100] <1>    mov     esi, [CCD_DriveDT]
406                   <1>
407                   <1> ; 19/12/2025
408                   <1> %if 0
409                   <1>    cmp     byte [Current_FATType], 1
410                   <1>    jnb    short loc_ccd_load_FS_sub_directory_next
411                   <1> %endif
412                   <1> ;push    esi
413 0000A4C7 E8691C0000    <1>    call    load_FAT_sub_directory
414                   <1> ;pop     edi ; Dos Drv Description Table
415                   <1>
416                   <1> pass_ccd_set_dir_cluster_ptr:
417                   <1> ;mov     edi, esi
418 0000A4CC BE[C07E0100]    <1>    mov     esi, PATH_Array
419 0000A4D1 7265         <1>    jc     short loc_ccd_retn_c
420                   <1>
421 0000A4D3 A1[8E7E0100]    <1>    mov     eax, [DirBuff_Cluster]
422                   <1>
423 0000A4D8 FE05[447F0100]    <1>    inc     byte [CCD_Level]
424 0000A4DE 0FB61D[447F0100] <1>    movzx   ebx, byte [CCD_Level]
425 0000A4E5 C0E304         <1>    shl     bl, 4    ; * 16 (<= 128)
426 0000A4E8 01DE         <1>    add     esi, ebx ; 19/02/2016
427 0000A4EA 89460C         <1>    mov     [esi+12], eax
428 0000A4ED EB1F         <1>    jmp     short loc_ccd_set_cdfc
429                   <1>
430                   <1> loc_ccd_load_FAT32_root_dir:
431 0000A4EF BE[C07E0100]    <1>    mov     esi, PATH_Array
432 0000A4F4 8B460C         <1>    mov     eax, [esi+12]
433                   <1>
434                   <1> loc_ccd_load_sub_directory: ; 19/12/2025
435 0000A4F7 8B35[407F0100] <1>    mov     esi, [CCD_DriveDT]
436                   <1>
437                   <1> loc_ccd_load_FAT_sub_directory:
438                   <1> ;push    esi
439 0000A4FD E8331C0000    <1>    call    load_FAT_sub_directory
440                   <1> ;pop     edi ; Dos Drv Description Table
441                   <1>
442                   <1> pass_ccd_load_FAT_sub_directory:
443                   <1> ;mov     edi, esi
444 0000A502 BE[C07E0100]    <1>    mov     esi, PATH_Array
445 0000A507 722F         <1>    jc     short loc_ccd_retn_c
446                   <1>
447 0000A509 A1[8E7E0100]    <1>    mov     eax, [DirBuff_Cluster]
448                   <1>
449                   <1> loc_ccd_set_cdfc:
450 0000A50E 8A0D[447F0100]    <1>    mov     cl, [CCD_Level]
451 0000A514 880D[60770100]    <1>    mov     [Current_Dir_Level], cl
452 0000A51A A3[5C770100]    <1>    mov     [Current_Dir_FCluster], eax
453                   <1>
454 0000A51F 8A2D[457F0100]    <1>    mov     ch, [Last_Dir_Level]
455 0000A525 38E9         <1>    cmp     cl, ch
456                   <1> ;jb     loc_ccd_load_child_dir_next
457                   <1> ; 28/07/2022
458 0000A527 7305         <1>    jnb    short loc_ccd_2
459 0000A529 E93CFFFFFF    <1>    jmp     loc_ccd_load_child_dir_next
460                   <1>
461 0000A52E 803D[4E7F0100]CD <1> loc_ccd_2:
462 0000A535 7408         <1>    cmp     byte [CD_COMMAND], 0CDh ; 'CD' command or another
463                   <1>    je     short loc_ccd_save_current_dir
464                   <1>
465                   <1> ; jne -> don't save, restore (the previous cdir) later !
466                   <1> ; (saving the cdir would prevent previous cdir restoration!)
467 0000A537 F8           <1>    clc
468                   <1>
469                   <1> loc_ccd_retn_c:
470 0000A538 8B3D[407F0100] <1>    mov     edi, [CCD_DriveDT]
471 0000A53E C3           <1>    retn
472                   <1>
473                   <1> ; 19/12/2025
474                   <1> %if 0
475                   <1> loc_ccd_load_sub_directory:
476                   <1>    mov     esi, [CCD_DriveDT]
477                   <1>    cmp     byte [Current_FATType], 1
478                   <1>    jnb    short loc_ccd_load_FAT_sub_directory

```

```

479      call    load_FS_sub_directory
480      jmp     short pass_ccd_load_FAT_sub_directory
481  %endif
482
483      loc_ccd_save_current_dir:
484      ; 02/03/2021 (TRDOS 386 v2.0.3) ((BugFix))
485      ; ('find_directory_entry' has been fixed to prevent large
486      ; ECX value > 65535)
487      ;
488      mov     esi, PATH_Array ; 19/02/2016
489      mov     edi, [CCD_DriveDT]
490      push    edi
491      add     edi, LD_CDirLevel
492      mov     [edi], cl
493      inc     edi ; LD_CurrentDirectory
494      push    esi
495      ; mov     ecx, 32 ; always < 65536 (in this procedure)
496      mov     cx, 32
497      ; 02/03/2021
498      ; mov     ecx, 32
499      rep     movsd
500      ; Current directory has been saved to
501      ; the DOS drive description table, cdir area !
502      pop     esi ; PATH_Array
503      pop     edi ; Dos Drv Description Table
504
505      retn
506
507      parse_dir_name:
508      ; 11/02/2016
509      ; 10/02/2016
510      ; 07/02/2016 (TRDOS 386 = TRDOS v2.0)
511      ; 18/09/2011
512      ; 17/10/2009
513      ; INPUT ->
514      ; ESI = ASCIIZ Directory String Address
515      ; AL = Current Directory Level
516      ; EDI = Destination Address
517      ; (8 levels, each one 12+4 byte)
518      ; OUTPUT ->
519      ; EDI = Dir Entry Formatted Array
520      ; with zero cluster pointer at the last level
521      ; AH = Last Dir Level
522      ; AL = Current Dir Level
523      ;
524      ; (esi, ebx, ecx will be changed)
525
526      ; mov     [PATH_Array_Ptr], edi
527      mov     ah, al
528      mov     [PATH_CDLevel], ax
529      repeat_ppdn_check_slash:
530      lodsb
531      cmp     al, '/'
532      je      short repeat_ppdn_check_slash
533      cmp     al, 21h
534      jnb     short loc_ppdn_retn
535      push    edi
536      loc_ppdn_get_dir_name:
537      mov     ecx, 12
538      mov     edi, Dir_File_Name
539      repeat_ppdn_get_dir_name:
540      stosb
541      lodsb
542      cmp     al, '/'
543      je      short loc_check_level_dot_conv_dir_name
544      cmp     al, 20h
545      jna     short loc_ppdn_end_of_path_scan
546      loop    repeat_ppdn_get_dir_name
547      pop     edi
548      stc
549      loc_ppdn_retn:
550      retn
551
552      loc_ppdn_end_of_path_scan:
553      dec     esi
554      loc_check_level_dot_conv_dir_name:
555      xor     eax, eax
556      stosb
557      mov     ebx, esi
558      mov     esi, Dir_File_Name
559      lodsb
560      repeat_ppdn_name_check_dot:
561      cmp     al, '.'
562      jne     short loc_ppdn_convert_sub_dir_name
563      repeat_ppdn_name_dot_dot:
564      lodsb
565      cmp     al, '.'
566      je      short loc_ppdn_dot_dot
567      cmp     al, 21h
568      jnb     short pass_ppdn_convert_sub_dir_name
569      loc_ppdn_convert_sub_dir_name:
570      mov     ah, [PATH_Level]
571      cmp     ah, 7
572      jnb     short pass_ppdn_convert_sub_dir_name
573      inc     ah
574      mov     [PATH_Level], ah
575      mov     esi, Dir_File_Name
576      ; mov     edi, [PATH_Array_Ptr]
577      mov     al, 16
578      mul     ah
579      mov     edi, [esp]
580      ; push    edi
581      add     edi, eax
582      call    convert_file_name
583      ; pop     edi
584      pass_ppdn_convert_sub_dir_name:
585      mov     esi, ebx
586      repeat_ppdn_check_last_slash:
587      lodsb
588      cmp     al, '/'
589      je      short repeat_ppdn_check_last_slash
590      cmp     al, 21h
591      jnb     short loc_ppdn_get_dir_name
592      end_of_parse_dir_name:
593      pop     edi
594      cmc
595      ; mov     al, [PATH_CDLevel]
596      ; mov     ah, [PATH_Level]
597      mov     ax, [PATH_CDLevel]
598      retn
599
600      loc_ppdn_dot_dot:
601      lodsb
602      cmp     al, 21h

```

```

603 0000A5DC 73F2      <1>      jnb      short end_of_parse_dir_name
604                  <1> loc_ppdn_dot_dot_prev_level:
605 0000A5DE 66A1[E47F0100] <1>      mov     ax, [PATH_CDLevel]
606 0000A5E4 80EC01      <1>      sub     ah, 1
607 0000A5E7 80D400      <1>      adc     ah, 0
608 0000A5EA 38E0      <1>      cmp     al, ah
609 0000A5EC 7602      <1>      jna     short pass_ppdn_set_al_to_ah
610 0000A5EE 88E0      <1>      mov     al, ah
611                  <1> pass_ppdn_set_al_to_ah:
612 0000A5F0 66A3[E47F0100] <1>      mov     [PATH_CDLevel], ax
613 0000A5F6 EBCD      <1>      jmp     short pass_ppdn_convert_sub_dir_name
614                  <1>
615                  <1> locate_current_dir_file:
616                  <1> ; 19/12/2025 (TRDOS 386 Kernel v2.0.10)
617                  <1> ; 26/08/2024 (TRDOS 386 v2.0.9)
618                  <1> ; 26/08/2024 (TRDOS 386 v2.0.9)
619                  <1> ; 28/07/2022 (TRDOS 386 Kernel v2.0.5)
620                  <1> ; 20/11/2017
621                  <1> ; 14/02/2016
622                  <1> ; 13/02/2016
623                  <1> ; 10/02/2016
624                  <1> ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
625                  <1> ; 14/08/2010
626                  <1> ; 19/09/2009
627                  <1> ; 2005
628                  <1> ; INPUT ->
629                  <1> ; ESI = DOS DirEntry Format FileName Address
630                  <1> ; AL = Attributes Mask
631                  <1> ; (<AL AND EntryAttrib> must be equal to AL)
632                  <1> ; AH = Negative Attributes Mask (If AH>0)
633                  <1> ; (<AH AND EntryAttrib> must be ZERO)
634                  <1> ; CH > 0 Find First Free Dir Entry or Deleted Entry
635                  <1> ; CL = 0 -> Return the First Free Dir Entry
636                  <1> ; CL = E5h -> Return the 1st deleted entry
637                  <1> ; CL = FFh -> Return the 1st deleted or free entry
638                  <1> ; CL > 0 and CL <> E5h and CL <> FFh -> Return the first
639                  <1> ; proper entry (which fits with Attributes Masks)
640                  <1> ; CX = 0 Find Valid File/Directory/VolumeName
641                  <1> ; ? = Any One Char
642                  <1> ; * = Every Chars
643                  <1> ; OUTPUT ->
644                  <1> ; EDI = Directory Entry Address (in Directory Buffer)
645                  <1> ; ESI = DOS DirEntry Format FileName Address
646                  <1> ; CF = 0 -> No Error, Proper Entry,
647                  <1> ; DL = Attributes
648                  <1> ; DH = Previous Entry Attr (LongName Check)
649                  <1> ; AL > 0 -> Ambiguous filename wildcard "?" used
650                  <1> ; AH > 0 -> Ambiguous filename wildcard "*" used
651                  <1> ; AX = 0 -> Filename full fits with directory entry
652                  <1> ; CH = The 1st Name Char of Current Dir Entry
653                  <1> ; CF = 1 -> Proper entry not found, Error Code in EAX/AL
654                  <1> ; CL = 0 and CH = 0 -> Free Entry (End Of Dir)
655                  <1> ; CL = 0 and CH = E5h -> Deleted Entry fits with filters
656                  <1> ; CL > 0 -> Entry not found, CH invalid
657                  <1> ; CF = 0 ->
658                  <1> ; EBX = Current Directory Entry Index/Number (BX)
659                  <1>
660                  <1> ;mov     word [DirBuff_EntryCounter], 0 ; Zero Based
661                  <1>
662 0000A5F8 8935[487F0100] <1>      mov     [CDLF_FNAddress], esi
663 0000A5FE 66A3[467F0100] <1>      mov     [CDLF_AttributesMask], ax
664 0000A604 66890D[4C7F0100] <1>      mov     [CDLF_DEType], cx
665                  <1>
666 0000A60B 31DB      <1>      xor     ebx, ebx
667 0000A60D 881D[5C7F0100] <1>      mov     [PreviousAttr], bl ; 0 ; 13/02/2016
668                  <1>
669 0000A613 8A3D[62770100] <1>      mov     bh, [Current_Drv]
670 0000A619 381D[897E0100] <1>      cmp     byte [DirBuff_ValidData], bl ; 0
671 0000A61F 761D      <1>      jna     short loc_lcdf_reload_current_dir2
672 0000A621 8A1D[877E0100] <1>      mov     bl, [DirBuff_DRV]
673 0000A627 80EB41      <1>      sub     bl, 'A'
674 0000A62A 38DF      <1>      cmp     bh, bl
675 0000A62C 750E      <1>      jne     short loc_lcdf_reload_current_dir1
676 0000A62E 8B15[8E7E0100] <1>      mov     edx, [DirBuff_Cluster]
677 0000A634 3B15[5C770100] <1>      cmp     edx, [Current_Dir_FCluster]
678 0000A63A 7412      <1>      je      short loc_cdir_locatefile_search
679                  <1>
680                  <1> loc_lcdf_reload_current_dir1:
681 0000A63C 30DB      <1>      xor     bl, bl
682                  <1> loc_lcdf_reload_current_dir2:
683 0000A63E 89DE      <1>      mov     esi, ebx
684 0000A640 81C600010900 <1>      add     esi, Logical_DOSDisks
685 0000A646 E866000000 <1>      call    reload_current_directory
686 0000A64B 734F      <1>      jnc     short loc_locatefile_search_again
687 0000A64D C3      <1>      retn
688                  <1>
689                  <1> loc_cdir_locatefile_search:
690 0000A64E 31DB      <1>      xor     ebx, ebx
691 0000A650 55      <1>      push    ebp ; 20/11/2017
692 0000A651 E87D000000 <1>      call    find_directory_entry
693 0000A656 5D      <1>      pop     ebp ; 20/11/2017
694 0000A657 733B      <1>      jnc     short loc_cdir_locate_file_retn
695                  <1>
696                  <1> loc_locatefile_check_stc_reason:
697 0000A659 08ED      <1>      or      ch, ch
698 0000A65B 7436      <1>      jz      short loc_cdir_locate_file_stc_retn
699                  <1>
700                  <1> loc_locatefile_check_next_entryblock:
701                  <1> ;mov     bh, [Current_Drv]
702                  <1> ;sub     bl, bl
703                  <1> ;movzx   esi, bx
704                  <1> ; 28/07/2022
705 0000A65D 31DB      <1>      xor     ebx, ebx
706 0000A65F 8A3D[62770100] <1>      mov     bh, [Current_Drv]
707 0000A665 89DE      <1>      mov     esi, ebx
708 0000A667 81C600010900 <1>      add     esi, Logical_DOSDisks
709                  <1>
710 0000A66D 803D[60770100]00 <1>      cmp     byte [Current_Dir_Level], 0
711                  <1> ; 19/12/2025
712                  <1> %if 0
713                  <1> jna     short loc_locatefile_check_FAT_type
714                  <1>
715                  <1> cmp     byte [Current_FATType], 1
716                  <1> jnb     short loc_locatefile_load_subdir_cluster
717                  <1> retn
718                  <1> %else
719                  <1> ; 19/12/2025
720 0000A674 7709      <1>      ja      short loc_locatefile_load_subdir_cluster
721                  <1> %endif
722                  <1>
723                  <1> loc_locatefile_check_FAT_type:
724 0000A676 803D[61770100]03 <1>      cmp     byte [Current_FATType], 3
725 0000A67D 7215      <1>      jb      short loc_cdir_locate_file_retn
726                  <1>

```

```

727 <1> loc_locatefile_load_subdir_cluster:
728 0000A67F A1[8E7E0100] <1> mov     eax, [DirBuff_Cluster]
729 0000A684 E8F0180000 <1> call    get_next_cluster
730 0000A689 730A <1> jnc     short loc_locatefile_next_cluster
731 0000A68B 09C0 <1> or      eax, eax
732 0000A68D 7504 <1> jnz     short loc_locatefile_drive_not_ready_read_err
733 <1> ; stc
734 <1> ; 28/07/2022
735 <1> ; loc_locatefile_file_notfound:
736 <1> ; mov     eax, 2 ; File/Directory/VolName not found
737 <1> ; xor     eax, eax
738 <1> ; 26/08/2024
739 <1> ; eax = 0
740 0000A68F B002 <1> mov     al, 2
741 0000A691 F9 <1> stc
742 0000A692 C3 <1> retn
743 <1>
744 <1> loc_locatefile_drive_not_ready_read_err:
745 <1> ; mov     eax, 17 ; Drive not ready or read error
746 <1> loc_cdir_locate_file_stc_retn:
747 0000A693 F5 <1> cmc ; stc
748 <1> loc_cdir_locate_file_retn:
749 0000A694 C3 <1> retn
750 <1>
751 <1> loc_locatefile_next_cluster:
752 0000A695 E89B1A0000 <1> call    load_FAT_sub_directory
753 <1> ; jc     short loc_locatefile_drive_not_ready_read_err
754 0000A69A 72F8 <1> jc      short loc_cdir_locate_file_retn
755 <1>
756 <1> loc_locatefile_search_again:
757 0000A69C 8B35[487F0100] <1> mov     esi, [CDLF_FNAddress]
758 0000A6A2 66A1[467F0100] <1> mov     ax, [CDLF_AttributesMask]
759 0000A6A8 668B0D[4C7F0100] <1> mov     cx, [CDLF_DEType]
760 0000A6AF EB9D <1> jmp     short loc_cdir_locatefile_search
761 <1>
762 <1> reload_current_directory:
763 <1> ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
764 <1> ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
765 <1> ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
766 <1> ; 13/06/2010
767 <1> ; 22/09/2009
768 <1> ;
769 <1> ; INPUT ->
770 <1> ; ESI = Dos drive description table address
771 <1>
772 <1> ; mov     al, [esi+LD_FATType]
773 0000A6B1 A0[61770100] <1> mov     al, [Current_FATType]
774 0000A6B6 3C02 <1> cmp     al, 2
775 0000A6B8 770F <1> ja      short loc_reload_FAT_sub_directory
776 0000A6BA 8A25[60770100] <1> mov     ah, [Current_Dir_Level]
777 <1> ; 19/12/2025
778 <1> %if 0
779 <1> or      al, al
780 <1> jz      short loc_reload_FS_directory
781 <1> %endif
782 0000A6C0 08E4 <1> or      ah, ah
783 0000A6C2 7505 <1> jnz     short loc_reload_FAT_sub_directory
784 <1> loc_reload_FAT_12_16_root_directory:
785 <1> ; call    load_FAT_root_directory
786 <1> ; retn
787 <1> ; 28/07/2022
788 0000A6C4 E9EE190000 <1> jmp     load_FAT_root_directory
789 <1>
790 <1> ; 19/12/2025
791 <1> %if 0
792 <1> loc_reload_FS_directory:
793 <1> and     ah, ah
794 <1> jnz     short loc_reload_FS_sub_directory
795 <1> loc_reload_FS_root_directory:
796 <1> ; call    load_FS_root_directory
797 <1> ; retn
798 <1> ; 28/07/2022
799 <1> jmp     load_FS_root_directory
800 <1> loc_reload_FS_sub_directory:
801 <1> mov     eax, [Current_Dir_FCluster]
802 <1> ; call    load_FS_sub_directory
803 <1> ; retn
804 <1> jmp     load_FS_sub_directory
805 <1> %endif
806 <1>
807 <1> loc_reload_FAT_sub_directory:
808 0000A6C9 A1[5C770100] <1> mov     eax, [Current_Dir_FCluster]
809 <1> ; call    load_FAT_sub_directory
810 <1> ; retn
811 <1> ; 28/07/2022
812 0000A6CE E9621A0000 <1> jmp     load_FAT_sub_directory
813 <1>
814 <1> find_directory_entry:
815 <1> ; 28/07/2022 (TRDOS 386 kernel v2.0.5)
816 <1> ; 02/03/2021 (TRDOS 386 v2.0.3) ((BugFix))
817 <1> ; 14/02/2016
818 <1> ; 13/02/2016
819 <1> ; 10/02/2016
820 <1> ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
821 <1> ; 14/08/2010 (DIR.ASM, "proc_find_direntry")
822 <1> ; 19/09/2009
823 <1> ; 2005
824 <1> ; INPUT ->
825 <1> ; ESI = Sub Dir or File Name Address
826 <1> ; AL = Attributes Mask
827 <1> ; (<AL AND EntryAttrib> must be equal to AL)
828 <1> ; AH = Negative Attributes Mask (If AH>0)
829 <1> ; (<AH AND EntryAttrib> must be ZERO)
830 <1> ; CH > 0 Find First Free Dir Entry or Deleted Entry
831 <1> ; CL = 0 -> Return the First Free Dir Entry
832 <1> ; CL = E5h -> Return the 1st deleted entry
833 <1> ; CL = FFh -> Return the 1st deleted or free entry
834 <1> ; CL > 0 and CL <> E5h and CL <> FFh -> Return the first
835 <1> ; proper entry (which fits with Attributes Masks)
836 <1> ; CX = 0 -> Find Valid File/Directory/VolumeName
837 <1> ; ? = Any One Char
838 <1> ; * = Every Chars
839 <1> ; EBX = Current Dir Entry (BX)
840 <1>
841 <1> ; OUTPUT ->
842 <1> ; EDI = Directory Entry Address (in DirectoryBuffer)
843 <1> ; ESI = Sub Dir or File Name Address
844 <1> ; CF = 0 -> No Error, Proper Entry,
845 <1> ; DL = Attributes
846 <1> ; DH = Previous Entry Attr (LongName Check)
847 <1> ; AL > 0 -> Ambiguous filename wildcard "?" used
848 <1> ; AH > 0 -> Ambiguous filename wildcard "*" used
849 <1> ; AX = 0 -> Filename full fits with directory entry
850 <1> ; EBX = CurrentDirEntry (BX)

```

```

851      <1>      ;      CH = The 1st Name Char of Current Dir Entry
852      <1>      ;      CF = 1 -> Proper entry not found, Error Code in AX/AL
853      <1>      ;      CL = 0 and CH = 0 -> Free Entry (End Of Dir)
854      <1>      ;      CL = 0 and CH = E5h -> Deleted Entry fits with filters
855      <1>      ;      CL > 0 -> Entry not found, CH invalid
856      <1>      ;
857      <1>      ; (EAX, EBX, ECX, EDX, EDI, EBP will be changed)
858      <1>
859      0000A6D3 663B1D[8C7E0100] <1>      cmp      bx, [DirBuff_LastEntry]
860      0000A6DA 7728 <1>      ja       short loc_ffde_stc_retn_255 ; 28/07/2022
861      <1>
862      <1>      ;mov      [DirBuff_CurrentEntry], bx
863      <1>
864      0000A6DC BF00000800 <1>      mov      edi, Directory_Buffer
865      0000A6E1 66A3[587F0100] <1>      mov      [FDE_AttrMask], ax
866      <1>
867      0000A6E7 29C0 <1>      sub      eax, eax
868      <1>
869      <1>      ;;mov      [PreviousAttr], al ; 0 ;; 13/02/2016
870      0000A6E9 66A3[5A7F0100] <1>      mov      [AmbiguousFileName], ax ; 0
871      <1>
872      0000A6EF 6689D8 <1>      mov      ax, bx
873      <1>      ;shl      ax, 5 ; ; * 32 ; Directory entry size
874      <1>      ; 28/07/2022
875      0000A6F2 C1E005 <1>      shl      eax, 5
876      0000A6F5 01C7 <1>      add      edi, eax
877      <1>
878      0000A6F7 08ED <1>      or       ch, ch
879      <1>      ;jnz      loc_find_free_deleted_entry_0
880      <1>      ; 28/07/2022
881      0000A6F9 7405 <1>      jz       short loc_fde_any_valid_entry_opt
882      0000A6FB E911010000 <1>      jmp      loc_find_free_deleted_entry_0
883      <1>
884      <1>      loc_fde_any_valid_entry_opt:
885      0000A700 08C9 <1>      or       cl, cl
886      <1>      ;jnz      loc_ffde_stc_retn_255
887      <1>      ; 28/07/2022
888      0000A702 742E <1>      jz       short check_find_dir_entry
889      <1>
890      <1>      ; 28/07/2022
891      <1>      loc_ffde_stc_retn_255:
892      <1>      ; 02/03/2021 (TRDOS 386 v2.0.3) ((BugFix))
893      <1>      ; (ECX must not be > 65535)
894      <1>      ; ((because 'loc_ccd_save_current_dir'
895      <1>      ; sets CX to 32 for 'rep movsd'))
896      0000A704 66B9FFFF <1>      mov      cx, 0FFFFh
897      <1>      ;xor      ecx, ecx
898      <1>      ;dec      ecx ; 0FFFFFFFh
899      <1>      ;xor      eax, eax
900      <1>      loc_find_direntry_stc_retn:
901      <1>      loc_check_ffde_retn_1:
902      <1>      ;mov      ax, 2
903      <1>      ;mov      eax, 2 ; File Not Found
904      <1>      ; 28/07/2022
905      0000A708 29C0 <1>      sub      eax, eax
906      0000A70A B002 <1>      mov      al, 2
907      0000A70C 8A35[5C7F0100] <1>      mov      dh, [PreviousAttr]
908      0000A712 66891D[8A7E0100] <1>      mov      [DirBuff_CurrentEntry], bx
909      0000A719 F9 <1>      stc
910      0000A71A C3 <1>      retn
911      <1>
912      <1>      ; 28/07/2022
913      <1>      loc_find_dir_next_entry_prevdeleted:
914      0000A71B 80CA80 <1>      or       dl, 80h ; Bit 7 -> deleted entry sign
915      <1>      ;jmp      short loc_find_dir_next_entry
916      <1>
917      <1>      ; 28/07/2022
918      <1>      loc_find_dir_next_entry:
919      0000A71E 8815[5C7F0100] <1>      mov      byte [PreviousAttr], dl ; LongName check
920      <1>      loc_find_dir_next_entry_1:
921      0000A724 5E <1>      pop      esi
922      0000A725 83C720 <1>      add      edi, 32
923      <1>      ;;inc      word [DirBuff_EntryCounter]
924      <1>      ;inc      bx
925      <1>      ; 28/07/2022
926      0000A728 43 <1>      inc      ebx
927      0000A729 663B1D[8C7E0100] <1>      cmp      bx, [DirBuff_LastEntry]
928      0000A730 77D2 <1>      ja       short loc_ffde_stc_retn_255
929      <1>      ; 28/07/2022
930      <1>      ;jmp      short check_find_dir_entry
931      <1>
932      <1>      check_find_dir_entry:
933      0000A732 66A1[587F0100] <1>      mov      ax, [FDE_AttrMask]
934      0000A738 8A2F <1>      mov      ch, [edi]
935      0000A73A 80FD00 <1>      cmp      ch, 0 ; Is it never used entry?
936      <1>      ;jna      loc_find_direntry_stc_retn
937      <1>      ; 28/07/2022
938      0000A73D 7702 <1>      ja       short loc_fde_check_attrib
939      <1>      ; end of directory entries
940      0000A73F EBC7 <1>      jmp      loc_find_direntry_stc_retn
941      <1>      loc_fde_check_attrib:
942      0000A741 56 <1>      push     esi
943      0000A742 8A570B <1>      mov      dl, [edi+0Bh] ; File attributes
944      0000A745 80FDE5 <1>      cmp      ch, 0E5h ; Is it a deleted file?
945      0000A748 74D1 <1>      je       short loc_find_dir_next_entry_prevdeleted
946      <1>
947      0000A74A 80FA0F <1>      cmp      dl, 0Fh ; longname sub component check
948      0000A74D 7505 <1>      jne      short loc_check_attributes_mask
949      0000A74F E8A6010000 <1>      call     save_longname_sub_component
950      <1>
951      <1>      loc_check_attributes_mask:
952      0000A754 88C6 <1>      mov      dh, al
953      0000A756 20D6 <1>      and      dh, dl
954      <1>      ; 28/07/2022
955      0000A758 38F0 <1>      cmp      al, dh
956      0000A75A 75C2 <1>      jne      short loc_find_dir_next_entry
957      0000A75C 20D4 <1>      and      ah, dl
958      0000A75E 75BE <1>      jnz      short loc_find_dir_next_entry
959      0000A760 80FA0F <1>      cmp      dl, 0Fh
960      0000A763 7515 <1>      jne      short pass_direntry_attr_check
961      <1>
962      0000A765 3C0F <1>      cmp      al, 0Fh ; AL = 0Fh -> find long name
963      0000A767 75B5 <1>      jne      short loc_find_dir_next_entry
964      <1>
965      0000A769 5E <1>      pop      esi
966      <1>      ;xor      ax, ax
967      <1>      ; 28/07/2022
968      <1>      ;sub      eax, eax
969      0000A76A 30C0 <1>      xor      al, al
970      0000A76C 8A35[5C7F0100] <1>      mov      dh, [PreviousAttr]
971      0000A772 66891D[8A7E0100] <1>      mov      [DirBuff_CurrentEntry], bx
972      0000A779 C3 <1>      retn
973      <1>
974      <1>      pass_direntry_attr_check:

```

```

975 0000A77A 89FD      <1>      mov     ebp, edi ; 14/02/2016
976                    <1>      ;mov     ecx, 8
977                    <1>      ; 28/07/2022
978 0000A77C 29C9      <1>      sub     ecx, ecx
979 0000A77E B108      <1>      mov     cl, 8
980                    <1>      loc_lodsb_find_dir:
981 0000A780 AC        <1>      lodsb
982 0000A781 3C2A      <1>      cmp     al, '*'
983 0000A783 7508      <1>      jne     short pass_fde_ambiguous1_check
984 0000A785 FE05[5B7F0100] <1>      inc     byte [AmbiguousFileName+1]
985 0000A78B EB23      <1>      jmp     short loc_check_direntry_extension
986                    <1>
987                    <1>      pass_fde_ambiguous1_check:
988 0000A78D 3C3F      <1>      cmp     al, '?'
989 0000A78F 750D      <1>      jne     short pass_fde_ambiguous2_check
990 0000A791 FE05[5A7F0100] <1>      inc     byte [AmbiguousFileName]
991 0000A797 803F20    <1>      cmp     byte [edi], 20h
992 0000A79A 763E      <1>      jna     short loc_find_dir_next_entry_ebp
993 0000A79C EB0F      <1>      jmp     short loc_scasb_find_dir_inc_di
994                    <1>
995                    <1>      pass_fde_ambiguous2_check:
996 0000A79E 3C20      <1>      cmp     al, 20h
997 0000A7A0 7507      <1>      jne     short loc_scasb_find_dir
998 0000A7A2 803F20    <1>      cmp     byte [edi], 20h
999 0000A7A5 7533      <1>      jne     short loc_find_dir_next_entry_ebp
1000 0000A7A7 EB07      <1>      jmp     short loc_check_direntry_extension
1001                    <1>
1002                    <1>      loc_scasb_find_dir:
1003 0000A7A9 3A07      <1>      cmp     al, [edi]
1004 0000A7AB 752D      <1>      jne     short loc_find_dir_next_entry_ebp
1005                    <1>      loc_scasb_find_dir_inc_di:
1006 0000A7AD 47        <1>      inc     edi
1007 0000A7AE E2D0      <1>      loop    loc_lodsb_find_dir
1008                    <1>
1009                    <1>      loc_check_direntry_extension:
1010 0000A7B0 BE08000000 <1>      mov     esi, 8
1011 0000A7B5 89F7      <1>      mov     edi, esi ; 8
1012 0000A7B7 033424    <1>      add     esi, [esp] ; Sub Dir or File Name Address
1013 0000A7BA 01EF      <1>      add     edi, ebp
1014 0000A7BC B103      <1>      mov     cl, 3
1015                    <1>      loc_lodsb_find_dir_ext:
1016 0000A7BE AC        <1>      lodsb
1017 0000A7BF 3C2A      <1>      cmp     al, '*'
1018 0000A7C1 7508      <1>      jne     short pass_fde_ambiguous3_check
1019 0000A7C3 FE05[5B7F0100] <1>      inc     byte [AmbiguousFileName+1]
1020 0000A7C9 EB1F      <1>      jmp     short loc_find_dir_proper_direntry
1021                    <1>
1022                    <1>      pass_fde_ambiguous3_check:
1023 0000A7CB 3C3F      <1>      cmp     al, '?'
1024 0000A7CD 7512      <1>      jne     short pass_fde_ambiguous4_check
1025 0000A7CF FE05[5A7F0100] <1>      inc     byte [AmbiguousFileName]
1026 0000A7D5 803F20    <1>      cmp     byte [edi], 20h
1027                    <1>      ;jna     short loc_find_dir_next_entry_ebp
1028                    <1>      ;jmp     short loc_scasb_find_dir_ext_inc_di
1029                    <1>      ; 28/07/2022
1030 0000A7D8 7732      <1>      ja     short loc_scasb_find_dir_ext_inc_di
1031                    <1>
1032                    <1>      loc_find_dir_next_entry_ebp:
1033 0000A7DA 89EF      <1>      mov     edi, ebp ; 14/02/2016
1034 0000A7DC E93DFFFFFF <1>      jmp     loc_find_dir_next_entry ; 28/07/2022
1035                    <1>
1036                    <1>      pass_fde_ambiguous4_check:
1037 0000A7E1 3C20      <1>      cmp     al, 20h
1038 0000A7E3 7523      <1>      jne     short loc_scasb_find_dir_ext
1039 0000A7E5 803F20    <1>      cmp     byte [edi], 20h
1040                    <1>      ; 28/07/2022
1041 0000A7E8 75F0      <1>      jne     short loc_find_dir_next_entry_ebp
1042                    <1>      ;jmp     short loc_find_dir_proper_direntry
1043                    <1>
1044                    <1>      loc_find_dir_proper_direntry:
1045 0000A7EA 30C9      <1>      xor     cl, cl
1046                    <1>      loc_find_dir_proper_direntry_1:
1047 0000A7EC 5E        <1>      pop     esi
1048 0000A7ED 89EF      <1>      mov     edi, ebp
1049 0000A7EF 8A2F      <1>      mov     ch, [edi]
1050 0000A7F1 8A570B    <1>      mov     dl, [edi+0Bh] ; Dir entry attributes
1051 0000A7F4 66A1[5A7F0100] <1>      mov     ax, [AmbiguousFileName]
1052                    <1>      loc_find_dir_proper_direntry_2:
1053 0000A7FA 8A35[5C7F0100] <1>      mov     dh, [PreviousAttr]
1054 0000A800 66891D[8A7E0100] <1>      mov     [DirBuff_CurrentEntry], bx
1055 0000A807 C3        <1>      retn
1056                    <1>
1057                    <1>      loc_scasb_find_dir_ext:
1058 0000A808 3A07      <1>      cmp     al, [edi]
1059 0000A80A 75CE      <1>      jne     short loc_find_dir_next_entry_ebp
1060                    <1>      loc_scasb_find_dir_ext_inc_di:
1061 0000A80C 47        <1>      inc     edi
1062 0000A80D E2AF      <1>      loop    loc_lodsb_find_dir_ext
1063 0000A80F EBD8      <1>      jmp     short loc_find_dir_proper_direntry_1
1064                    <1>
1065                    <1>      loc_find_free_deleted_entry_0:
1066 0000A811 66A1[587F0100] <1>      mov     ax, [FDE_AttrMask]
1067 0000A817 8A2F      <1>      mov     ch, [edi]
1068 0000A819 8A570B    <1>      mov     dl, [edi+0Bh] ; File attributes
1069 0000A81C 08C9      <1>      or      cl, cl
1070 0000A81E 7407      <1>      jz     short loc_check_ffde_0_repeat
1071                    <1>      ;cmp     cl, 0E5h
1072                    <1>      ;je     short pass_loc_check_ffde_0_err
1073 0000A820 80F9FF    <1>      cmp     cl, 0FFh
1074 0000A823 7430      <1>      je     short loc_find_free_deleted_entry_1
1075 0000A825 EB4A      <1>      jmp     short pass_loc_check_ffde_0_err
1076                    <1>
1077                    <1>      loc_check_ffde_0_repeat:
1078 0000A827 08ED      <1>      or      ch, ch
1079 0000A829 7510      <1>      jnz     short loc_check_ffde_0_next
1080                    <1>
1081                    <1>      loc_check_ffde_retn_2:
1082                    <1>      ;sub     ax, ax
1083                    <1>      ; 28/07/2022
1084 0000A82B 29C0      <1>      sub     eax, eax
1085 0000A82D 8A35[5C7F0100] <1>      mov     dh, [PreviousAttr]
1086 0000A833 66891D[8A7E0100] <1>      mov     [DirBuff_CurrentEntry], bx
1087 0000A83A C3        <1>      retn
1088                    <1>
1089                    <1>      loc_check_ffde_0_next:
1090                    <1>      ;inc     bx
1091                    <1>      ; 28/07/2022
1092 0000A83B 43        <1>      inc     ebx
1093 0000A83C 83C720    <1>      add     edi, 32
1094                    <1>      ;inc     word [DirBuff_EntryCounter]
1095                    <1>
1096 0000A83F 663B1D[8C7E0100] <1>      cmp     bx, [DirBuff_LastEntry]
1097                    <1>      ;ja     short loc_ffde_stc_retn_255
1098                    <1>      ; 07/08/2022

```



```

1099 0000A846 773A      <1>      ja      short jmp_ffde_stc_retn_255
1100                                <1>
1101 0000A848 8815[5C7F0100] <1>      mov      [PreviousAttr], dl
1102 0000A84E 8A2F      <1>      mov      ch, [edi]
1103 0000A850 8A570B    <1>      mov      dl, [edi+0Bh] ; file attributes
1104 0000A853 EBD2      <1>      jmp      short loc_check_ffde_0_repeat
1105                                <1>
1106                                <1> loc_find_free_deleted_entry_1:
1107 0000A855 28D2      <1>      sub      dl, dl
1108                                <1> loc_find_free_deleted_entry_2:
1109 0000A857 20ED      <1>      and      ch, ch
1110 0000A859 74D0      <1>      jz       short loc_check_ffde_retn_2
1111 0000A85B 80FDE5    <1>      cmp      ch, 0E5h
1112 0000A85E 74CB      <1>      je       short loc_check_ffde_retn_2
1113                                <1>      ;inc      bx
1114                                <1>      ; 28/07/2022
1115 0000A860 43        <1>      inc      ebx
1116 0000A861 83C720    <1>      add      edi, 32
1117 0000A864 663B1D[8C7E0100] <1>      cmp      bx, [DirBuff_LastEntry]
1118                                <1>      ;ja      short loc_ffde_stc_retn_255
1119                                <1>      ; 07/08/2022
1120 0000A86B 7715      <1>      ja       short jmp_ffde_stc_retn_255
1121                                <1>
1122 0000A86D 8A2F      <1>      mov      ch, [edi]
1123 0000A86F EBE6      <1>      jmp      short loc_find_free_deleted_entry_2
1124                                <1>
1125                                <1> pass_loc_check_ffde_0_err:
1126 0000A871 38CD      <1>      cmp      ch, cl
1127 0000A873 741F      <1>      je       short loc_check_ffde_attr
1128                                <1>
1129                                <1>      ;inc      bx
1130                                <1>      ; 28/07/2022
1131 0000A875 43        <1>      inc      ebx
1132 0000A876 83C720    <1>      add      edi, 32
1133 0000A879 663B1D[8C7E0100] <1>      cmp      bx, [DirBuff_LastEntry]
1134                                <1>      ;ja      loc_ffde_stc_retn_255
1135                                <1>      ; 28/07/2022
1136 0000A880 7605      <1>      jna      short loc_ffde_save_prev_attr
1137                                <1> jmp_ffde_stc_retn_255: ; 07/08/2022
1138 0000A882 E97DFEFFFF <1>      jmp      loc_ffde_stc_retn_255
1139                                <1>
1140                                <1> loc_ffde_save_prev_attr: ; 28/07/2022
1141 0000A887 8815[5C7F0100] <1>      mov      [PreviousAttr], dl
1142 0000A88D 8A2F      <1>      mov      ch, [edi]
1143 0000A88F 8A570B    <1>      mov      dl, [edi+0Bh]
1144 0000A892 EBDD      <1>      jmp      short pass_loc_check_ffde_0_err
1145                                <1>
1146                                <1> loc_check_ffde_attr:
1147 0000A894 88C6      <1>      mov      dh, al
1148 0000A896 20D6      <1>      and      dh, dl
1149 0000A898 38F0      <1>      cmp      al, dh
1150 0000A89A 759F      <1>      jne      short loc_check_ffde_0_next
1151 0000A89C 20D4      <1>      and      ah, dl
1152 0000A89E 759B      <1>      jnz      short loc_check_ffde_0_next
1153 0000A8A0 30C9      <1>      xor      cl, cl
1154 0000A8A2 EB87      <1>      jmp      short loc_check_ffde_retn_2
1155                                <1>
1156                                <1> convert_file_name:
1157                                <1>      ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
1158                                <1>      ; 06/03/2016
1159                                <1>      ; 11/02/2016
1160                                <1>      ; 07/02/2016 (TRDOS 386 = TRDOS v2.0)
1161                                <1>      ; 06/10/2009
1162                                <1>      ; 2005
1163                                <1>      ;
1164                                <1>      ; INPUT ->
1165                                <1>      ; ESI = Dot File Name Location
1166                                <1>      ; EDI = Dir Entry Format File Name Location
1167                                <1>      ; OUTPUT ->
1168                                <1>      ; EDI = Dir Entry Format File Name Location
1169                                <1>      ; ESI = Dot File Name Location (capitalized)
1170                                <1>      ;
1171                                <1>      ; (ECX, AL will be changed)
1172                                <1>
1173 0000A8A4 56        <1>      push     esi
1174 0000A8A5 57        <1>      push     edi
1175                                <1>
1176                                <1>      ;mov      ecx, 11
1177                                <1>      ; 29/07 2022
1178 0000A8A6 29C9      <1>      sub      ecx, ecx
1179 0000A8A8 B10B      <1>      mov      cl, 11
1180 0000A8AA B020      <1>      mov      al, 20h
1181 0000A8AC F3AA      <1>      rep      stosb
1182                                <1>
1183 0000A8AE 8B3C24    <1>      mov      edi, [esp]
1184                                <1>
1185 0000A8B1 B10C      <1>      mov      cl, 12 ; file name length (max.)
1186                                <1>      ; 06/03/2016
1187                                <1>      ; Directory entry name limit (11 bytes) check for
1188                                <1>      ; 'rename_directory_entry' procedure.
1189                                <1>      ; (EDI points to Directory Entry)
1190                                <1>      ; (If the file name would not contain a dot
1191                                <1>      ; and file name length would be 12, this would cause to
1192                                <1>      ; overwrite the attributes byte of the directory entry.)
1193                                <1>      ;
1194 0000A8B3 B50B      <1>      mov      ch, 11 ; directory entry's name length
1195                                <1> loc_check_first_dot:
1196 0000A8B5 8A06      <1>      mov      al, [esi]
1197 0000A8B7 3C2E      <1>      cmp      al, 2Eh
1198 0000A8B9 750C      <1>      jne      short pass_check_first_dot
1199 0000A8BB 8807      <1>      mov      [edi], al
1200 0000A8BD 47        <1>      inc      edi
1201 0000A8BE 46        <1>      inc      esi
1202 0000A8BF FEC9      <1>      dec      cl
1203 0000A8C1 75F2      <1>      jnz      short loc_check_first_dot
1204                                <1>      ;;(ecx <= 12)
1205                                <1>      ;;loop loc_check_first_dot
1206 0000A8C3 EB30      <1>      jmp      short stop_convert_file
1207                                <1>
1208                                <1> loc_get_fchar:
1209 0000A8C5 8A06      <1>      mov      al, [esi]
1210                                <1> pass_check_first_dot:
1211 0000A8C7 3C61      <1>      cmp      al, 61h ; 'a'
1212 0000A8C9 7208      <1>      jb       short pass_name_capitalize
1213 0000A8CB 3C7A      <1>      cmp      al, 7Ah ; 'z'
1214 0000A8CD 7704      <1>      ja       short pass_name_capitalize
1215 0000A8CF 24DF      <1>      and      al, 0DFh
1216 0000A8D1 8806      <1>      mov      [esi], al
1217                                <1> pass_name_capitalize:
1218 0000A8D3 3C21      <1>      cmp      al, 21h
1219 0000A8D5 721E      <1>      jb       short stop_convert_file
1220 0000A8D7 3C2E      <1>      cmp      al, 2Eh ; '.'
1221 0000A8D9 750C      <1>      jne      short pass_dot_space
1222                                <1> add_dot_space:

```

```

1223 0000A8DB 80F904 <1>      cmp     cl, 4
1224 0000A8DE 760E <1>      jna     short inc_and_loop
1225 0000A8E0 47 <1>      inc     edi
1226 0000A8E1 FECD <1>      dec     ch ; 06/03/2016
1227 0000A8E3 FEC9 <1>      dec     cl
1228 0000A8E5 EBF4 <1>      jmp     short add_dot_space
1229 <1>
1230 <1>      ;mov     al, 4
1231 <1>      ;cmp     cl, al
1232 <1>      ;jna     short inc_and_loop
1233 <1>      ;sub     cl, al
1234 <1>      ;add     edi, ecx
1235 <1>      ;mov     cl, al
1236 <1>      ;jmp     short inc_and_loop
1237 <1>
1238 <1> pass_dot_space:
1239 0000A8E7 8807 <1>      mov     [edi], al
1240 <1> loc_after_double_dot:
1241 <1>      ; 06/03/2016
1242 0000A8E9 FECD <1>      dec     ch ; count down for 11 bytes dir entry limit
1243 0000A8EB 740A <1>      jz      short stop_convert_file_x
1244 0000A8ED 47 <1>      inc     edi
1245 <1> inc_and_loop:
1246 0000A8EE FEC9 <1>      dec     cl ; count down for 12 bytes filename limit
1247 0000A8F0 7403 <1>      jz      short stop_convert_file
1248 0000A8F2 46 <1>      inc     esi
1249 <1>      ;;(ecx <= 12)
1250 <1>      ;;;loop  loc_get_fchar
1251 0000A8F3 EBD0 <1>      jmp     short loc_get_fchar
1252 <1>
1253 <1> stop_convert_file:
1254 <1>      ; 06/03/2016
1255 0000A8F5 30ED <1>      xor     ch, ch
1256 <1>      ; ECX < 256 ; 'find_first_file' -> xor cl, cl
1257 <1> stop_convert_file_x:
1258 0000A8F7 5F <1>      pop     edi
1259 0000A8F8 5E <1>      pop     esi
1260 0000A8F9 C3 <1>      retn
1261 <1>
1262 <1> save_longname_sub_component:
1263 <1>      ; 13/02/2016
1264 <1>      ; 06/02/2016 (TRDOS 386 = TRDOS v2.0)
1265 <1>      ; 28/02/2010
1266 <1>      ; 17/10/2009
1267 <1>      ; INPUT ->
1268 <1>      ; EDI = Directory Entry
1269 <1>      ; // This procedure is called
1270 <1>      ; // from 'find_directory_entry' procedure.
1271 <1>      ; // If the last entry returns with
1272 <1>      ; // a non-zero LongnameFound value and
1273 <1>      ; // if LFN_CheckSum value is equal to
1274 <1>      ; // the next shortname checksum,
1275 <1>      ; // long name is valid.
1276 <1>      ; // If a longname is longer than 65 bytes,
1277 <1>      ; // it is invalid for trdos. (>45h)
1278 <1>
1279 0000A8FA 57 <1>      push    edi
1280 0000A8FB 56 <1>      push    esi
1281 <1>      ;push    ebx
1282 <1>      ;push    ecx
1283 <1>      ;push    edx
1284 0000A8FC 50 <1>      push    eax
1285 <1>
1286 0000A8FD 29C9 <1>      sub     ecx, ecx
1287 <1>      ;sub     eax, eax
1288 0000A8FF B11A <1>      mov     cl, 26
1289 <1>
1290 0000A901 0FB607 <1>      movzx   eax, byte [edi] ; LDIR_Order
1291 0000A904 3C41 <1>      cmp     al, 41h ; 40h (last long entry sign) + 1
1292 0000A906 722B <1>      jb      short pass_pslnsc_last_long_entry
1293 <1>
1294 0000A908 88C4 <1>      mov     ah, al
1295 0000A90A 80EC40 <1>      sub     ah, 40h
1296 0000A90D 8825[5E7F0100] <1>      mov     [LFN_EntryLength], ah
1297 <1>
1298 0000A913 3C45 <1>      cmp     al, 45h ; 40h (last long entry sign) + 5
1299 <1>      ; Max 130 byte length is usable in TRDOS
1300 <1>      ; 26*5 = 130
1301 0000A915 7753 <1>      ja      short loc_pslnsc_retn
1302 <1>
1303 0000A917 2407 <1>      and     al, 07h ; 0Fh
1304 0000A919 A2[5D7F0100] <1>      mov     [LongNameFound], al
1305 <1>
1306 0000A91E FEC8 <1>      dec     al
1307 <1>      ;mov     cl, 26
1308 0000A920 F6E1 <1>      mul     cl
1309 <1>
1310 0000A922 89C6 <1>      mov     esi, eax
1311 0000A924 01CE <1>      add     esi, ecx
1312 <1>      ; to make is an ASCIIIZ string
1313 <1>      ; with ax+26 bytes length
1314 0000A926 81C6[607F0100] <1>      add     esi, LongFileName
1315 0000A92C 66C7060000 <1>      mov     word [esi], 0
1316 0000A931 EB16 <1>      jmp     short loc_pslnsc_move_ldir_name2
1317 <1>
1318 <1> pass_pslnsc_last_long_entry:
1319 0000A933 3C04 <1>      cmp     al, 04h
1320 0000A935 7733 <1>      ja      short loc_pslnsc_retn
1321 0000A937 FE0D[5D7F0100] <1>      dec     byte [LongNameFound]
1322 0000A93D 3A05[5D7F0100] <1>      cmp     al, [LongNameFound]
1323 0000A943 7525 <1>      jne     short loc_pslnsc_retn
1324 <1>
1325 <1> loc_pslnsc_move_ldir_name1:
1326 0000A945 FEC8 <1>      dec     al
1327 <1>      ;mov     cl, 26
1328 0000A947 F6E1 <1>      mul     cl
1329 <1>
1330 <1> loc_pslnsc_move_ldir_name2:
1331 0000A949 8A4F0D <1>      mov     cl, [edi+0Dh] ; long name checksum
1332 0000A94C 880D[5F7F0100] <1>      mov     [LFN_CheckSum], cl
1333 0000A952 89FE <1>      mov     esi, edi ; LDIR_Order
1334 0000A954 BF[607F0100] <1>      mov     edi, LongFileName
1335 0000A959 01C7 <1>      add     edi, eax
1336 0000A95B 46 <1>      inc     esi
1337 0000A95C B105 <1>      mov     cl, 5 ; chars 1 to 5
1338 0000A95E F366A5 <1>      rep     movsw
1339 0000A961 83C603 <1>      add     esi, 3
1340 0000A964 A5 <1>      movsd   ; char 6 & 7
1341 0000A965 A5 <1>      movsd   ; char 8 & 9
1342 0000A966 A5 <1>      movsd   ; char 10 & 11
1343 0000A967 46 <1>      inc     esi
1344 0000A968 46 <1>      inc     esi
1345 0000A969 A5 <1>      movsd   ; char 12 & 13
1346 <1>

```

```

1347 <1> loc_pslnsc_retn:
1348 0000A96A 58 <1> pop eax
1349 <1> ;pop edx
1350 <1> ;pop ecx
1351 <1> ;pop ebx
1352 0000A96B 5E <1> pop esi
1353 0000A96C 5F <1> pop edi
1354 <1>
1355 0000A96D C3 <1> retn
1356 <1>
1357 <1> parse_path_name:
1358 <1> ; 02/12/2025 (TRDOS 386 v2.0.10, BugFix)
1359 <1> ; 03/09/2024 (TRDOS 386 v2.0.9)
1360 <1> ; 09/08/2022
1361 <1> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
1362 <1> ; 10/02/2016
1363 <1> ; 08/02/2016 (TRDOS 386 = TRDOS v2.0)
1364 <1> ; 10/009/2011 ('proc_parse_pathname')
1365 <1> ; 27/11/2009
1366 <1> ; 05/12/2004
1367 <1> ;
1368 <1> ; INPUT ->
1369 <1> ; ESI = Beginning of ASCIIZ pathname string
1370 <1> ; EDI = Destination Address
1371 <1> ; (which is TR-DOS FindFile data buffer)
1372 <1> ; OUTPUT ->
1373 <1> ; CF = 1 -> Error
1374 <1> ; EAX = Error Code (AL)
1375 <1> ;
1376 <1> ; (Modified registers: eax, ecx, esi, edi)
1377 <1>
1378 <1> ; Clear the pathname bytes in TR-DOS Findfile data buffer
1379 0000A96E 57 <1> push edi
1380 <1> ;mov ecx, 20 ; 80 bytes
1381 <1> ; 29/07/2022
1382 0000A96F 29C9 <1> sub ecx, ecx
1383 0000A971 B114 <1> mov cl, 20
1384 0000A973 31C0 <1> xor eax, eax
1385 0000A975 F3AB <1> rep stosd
1386 0000A977 5F <1> pop edi
1387 <1>
1388 0000A978 668B06 <1> mov ax, [esi]
1389 0000A97B 80FC3A <1> cmp ah, ':'
1390 0000A97E 741C <1> je short loc_ppn_change_drive
1391 0000A980 A0[62770100] <1> mov al, [Current_Drv]
1392 0000A985 EB33 <1> jmp short pass_ppn_change_drive
1393 <1>
1394 <1> pass_ppn_cdir:
1395 0000A987 8B35[82800100] <1> mov esi, [First_Path_Pos]
1396 0000A98D AC <1> lodsb
1397 <1>
1398 0000A98E 83C741 <1> loc_ppn_get_filename:
1399 <1> add edi, 65 ; FindFile_Name location
1400 <1> ; TRDOS Filename length must not be more than 12 bytes
1401 0000A991 B10C <1> ;mov ecx, 12
1402 <1> mov cl, 12
1403 0000A993 AA <1> loc_ppn_get_fnchar_next:
1404 0000A994 AC <1> stosb
1405 0000A995 3C21 <1> lodsb
1406 0000A997 7269 <1> cmp al, 21h
1407 0000A999 E2F8 <1> jb short loc_ppn_clc_return
1408 <1> loop loc_ppn_get_fnchar_next
1409 0000A99B C3 <1> loc_ppn_return:
1410 <1> retn
1411 <1>
1412 <1> loc_ppn_change_drive:
1413 <1> ; 29/07/2022
1414 0000A99C 24DF <1> ; ecx = 0
1415 0000A99E 2C41 <1> and al, 0DFh
1416 0000A9A0 7264 <1> sub al, 'A' ; A:
1417 0000A9A2 3805[7D2E0100] <1> jc short loc_ppn_invalid_drive
1418 0000A9A8 725C <1> cmp [Last_DOS_DiskNo], al
1419 <1> jb short loc_ppn_invalid_drive
1420 0000A9AA 46 <1>
1421 0000A9AB 46 <1> inc esi
1422 0000A9AC 8A26 <1> inc esi
1423 0000A9AE 80FC21 <1> mov ah, [esi]
1424 0000A9B1 7307 <1> cmp ah, 21h
1425 <1> jnb short pass_ppn_change_drive
1426 <1>
1427 <1> loc_ppn_cmd_failed:
1428 0000A9B3 8807 <1> ; File or directory name is not existing
1429 0000A9B5 66B80100 <1> mov [edi], al ; Drv
1430 <1> mov ax, 1 ; eax = 1
1431 <1> ; TR-DOS Error Code 01h = Bad Command Argument
1432 <1> ; MS-DOS Error Code 01h : Invalid Function Number
1433 <1> ;stc
1434 0000A9B9 C3 <1> ; (MainProg ErrMsg: "Bad command or file name!")
1435 <1> retn
1436 <1>
1437 0000A9BA 8935[82800100] <1> pass_ppn_change_drive:
1438 <1> mov [First_Path_Pos], esi
1439 <1> ;mov dword [Last_Slash_Pos], 0
1440 0000A9C0 890D[86800100] <1> ; 29/07/2022
1441 0000A9C6 AA <1> mov [Last_Slash_Pos], ecx ; 0
1442 0000A9C7 8A06 <1> stosb
1443 <1> mov al, [esi]
1444 0000A9C9 3C2F <1> loc_scan_ppn_dslash:
1445 0000A9CB 7506 <1> cmp al, '/'
1446 0000A9CD 8935[86800100] <1> jne short loc_scan_next_slash_pos
1447 <1> mov [Last_Slash_Pos], esi
1448 0000A9D3 46 <1> loc_scan_next_slash_pos:
1449 0000A9D4 8A06 <1> inc esi
1450 0000A9D6 3C20 <1> mov al, [esi]
1451 0000A9D8 77EF <1> cmp al, 20h
1452 <1> ja short loc_scan_ppn_dslash
1453 <1> ;cmp dword [Last_Slash_Pos], 0
1454 <1> ; 09/08/2022
1455 <1> ;cmp [Last_Slash_Pos], ecx ; 0 ?
1456 <1> ;jna short pass_ppn_cdir
1457 0000A9DA 8B0D[86800100] <1> mov ecx, [Last_Slash_Pos]
1458 <1> ; 03/09/2024
1459 <1> ;jcxz pass_ppn_cdir
1460 <1> ; 02/12/2025
1461 0000A9E0 E3A5 <1> jecxz pass_ppn_cdir
1462 <1>
1463 0000A9E2 8B35[82800100] <1> mov esi, [First_Path_Pos]
1464 0000A9E8 29F1 <1> sub ecx, esi
1465 0000A9EA 41 <1> inc ecx
1466 <1> ;cmp ecx, 64
1467 0000A9EB 80F940 <1> cmp cl, 64
1468 0000A9EE 7715 <1> ja short loc_ppn_invalid_drive_stc
1469 <1>
1470 0000A9F0 89F8 <1> mov eax, edi ; Dest Dir String Location (65 bytes)

```

```

1471 0000A9F2 F3A4      <1>      rep      movsb
1472                    <1>      ;mov      [edi], cl ; 0, End of Dir String
1473 0000A9F4 8B35[86800100] <1>      mov      esi, [Last_Slash_Pos]
1474 0000A9FA 46          <1>      inc      esi
1475 0000A9FB 89C7          <1>      mov      edi, eax
1476 0000A9FD AC          <1>      lodsb
1477 0000A9FE 3C21          <1>      cmp      al, 21h
1478 0000AA00 738C          <1>      jnb      short loc_ppn_get_filename
1479                    <1>      loc_ppn_clc_return:
1480                    <1>      ;clc
1481 0000AA02 31C0          <1>      xor      eax, eax
1482 0000AA04 C3          <1>      retn
1483                    <1>
1484                    <1>      loc_ppn_invalid_drive_stc:
1485 0000AA05 F5          <1>      cmc      ; stc
1486                    <1>      loc_ppn_invalid_drive:
1487                    <1>      ; cf = 1
1488                    <1>      ; The Drive Letter/Char < "A" or > "Z"
1489 0000AA06 66B80F00      <1>      mov      ax, 0Fh
1490                    <1>      ; MS-DOS Error Code 0Fh = Disk Drive Invalid
1491                    <1>      ; (MainProg ErrMsg: "Drive not ready or read error!")
1492 0000AA0A C3          <1>      retn
1493                    <1>
1494                    <1>      find_longname:
1495                    <1>      ; 19/12/2025 (TRDOS 386 v2.0.10)
1496                    <1>      ; 29/07/2022 (TRDOS 386 Kernel v2.0.5)
1497                    <1>      ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)
1498                    <1>      ; 24/01/2010 (DIR.ASM, 'proc_find_longname')
1499                    <1>      ; 17/10/2009
1500                    <1>
1501                    <1>      ; INPUT ->
1502                    <1>      ; ESI = DOS short file name address
1503                    <1>      ; for example: "filename.ext"
1504                    <1>
1505                    <1>      ; OUTPUT ->
1506                    <1>      ; ESI = ASCIIZ longname address (cf = 0)
1507                    <1>      ; cf = 1 -> error number returns in EAX (AL)
1508                    <1>      ; AL = 0 & CF=1 -> longname not found
1509                    <1>      ; the file/directory has no longname
1510                    <1>      ; cf = 0 -> AL = FAT Type
1511                    <1>
1512                    <1>      ; 17/10/2009
1513                    <1>      ; ASCIIZ string will be returned
1514                    <1>      ; as LongFileName
1515                    <1>      ; clearing/reset is not needed
1516                    <1>      ;mov      ecx, 33
1517                    <1>      ;mov      edi, LongFileName
1518                    <1>      ;sub      ax, ax ; 0
1519                    <1>      ;rep      stosw
1520                    <1>
1521                    <1>      ;mov      byte [LongNameFound], 0
1522                    <1>
1523                    <1>      ; ESI = ASCIIZ file/directory name address
1524                    <1>      ; AL = Attributes AND mask
1525                    <1>      ; (Result of AND must be equal to AL)
1526                    <1>      ; AH = Negative attributes mask
1527                    <1>      ; (Result of AND must be ZERO)
1528 0000AA0B 66B80008      <1>      mov      ax, 0800h
1529                    <1>      ; it must not be volume name or longname
1530 0000AA0F E856DFFFFF      <1>      call     find_first_file
1531 0000AA14 7207          <1>      jc      short loc_fln_retn
1532                    <1>
1533                    <1>      ; 19/12/2025
1534                    <1>      %if 0
1535                    <1>      loc_fln_check_FAT_Type:
1536                    <1>      cmp      byte [Current_FATType], 1
1537                    <1>      jnb      short loc_fln_check_longname_yes_sign
1538                    <1>
1539                    <1>      ;call     get_fs_longname
1540                    <1>      ;retn
1541                    <1>      ; 29/07/2022
1542                    <1>      jmp      get_fs_longname
1543                    <1>      %endif
1544                    <1>
1545                    <1>      loc_fln_check_longname_yes_sign:
1546 0000AA16 08FF          <1>      or      bh, bh
1547 0000AA18 7504          <1>      jnz      short loc_fln_check_longnamefound_number
1548                    <1>      loc_fln_longname_not_found_retn:
1549 0000AA1A 31C0          <1>      xor      eax, eax
1550                    <1>      ; cf = 1 & al = 0 -> longname not found
1551 0000AA1C F9          <1>      stc
1552                    <1>      loc_fln_retn:
1553 0000AA1D C3          <1>      retn
1554                    <1>
1555                    <1>      loc_fln_check_longnamefound_number:
1556                    <1>      ; 'LongNameFound' is set by
1557                    <1>      ; ; by 'save_longname_sub_component'
1558                    <1>      ; which is called from
1559                    <1>      ; 'find_directory_entry'
1560                    <1>      ; which is called from
1561                    <1>      ; 'find_first_file'
1562                    <1>      ; It must 1 if the longname is valid
1563 0000AA1E 803D[5D7F0100]01 <1>      cmp      byte [LongNameFound], 1
1564 0000AA25 75F3          <1>      jne      short loc_fln_longname_not_found_retn
1565                    <1>
1566                    <1>      loc_fln_calculate_checksum:
1567 0000AA27 E813000000      <1>      call     calculate_checksum
1568                    <1>      ; AL = shortname checksum
1569                    <1>
1570                    <1>      loc_fln_longname_validation:
1571                    <1>      ; 'LFN_CheckSum' has been set already
1572                    <1>      ; by 'save_longname_sub_component'
1573                    <1>      ; which is called from
1574                    <1>      ; 'find_directory_entry'
1575                    <1>      ; which is called from
1576                    <1>      ; 'find_first_file'
1577 0000AA2C 3805[5F7F0100] <1>      cmp      [LFN_CheckSum], al
1578 0000AA32 75E6          <1>      jne      short loc_fln_longname_not_found_retn
1579                    <1>
1580 0000AA34 BE[607F0100] <1>      mov      esi, LongFileName
1581 0000AA39 A0[61770100] <1>      mov      al, [Current_FATType]
1582 0000AA3E C3          <1>      retn
1583                    <1>
1584                    <1>      calculate_checksum:
1585                    <1>      ; 29/07/2022 (TRDOS 386 Kernel v2.0.5)
1586                    <1>      ; 13/02/2016 (TRDOS 386 = TRDOS v2.0)
1587                    <1>      ; 17/10/2009 (DIR.ASM, 'proc_calculate_checksum')
1588                    <1>      ;
1589                    <1>      ; INPUT ->
1590                    <1>      ; ESI = 11 byte DOS File Name location
1591                    <1>      ; (in DOS Directory Entry Format)
1592                    <1>      ; OUTPUT ->
1593                    <1>      ; AL = 8 bit checksum (CRC) value
1594                    <1>      ;

```

```

1595      <1>      ; (Modified registers: EAX, ECX, ESI)
1596      <1>
1597      <1>      ; Erdogan Tan [ 17-10-2009 ]
1598      <1>      ; 'ror al, 1' instruction
1599      <1>
1600      <1>      ; Erdogan Tan [ 20-06-2004 ]
1601      <1>      ; This 8086 assembly code is an original code
1602      <1>      ; which is adapted from C code in
1603      <1>      ; Microsoft FAT32 File System Specification
1604      <1>      ; Version 1.03, December 6, 2000
1605      <1>      ; Page 28
1606      <1>
1607      0000AA3F 30C0      <1>      xor     al, al
1608      <1>      ;mov     ecx, 11
1609      <1>      ; 29/07/2022
1610      0000AA41 29C9      <1>      sub     ecx, ecx
1611      0000AA43 B10B      <1>      mov     cl, 11
1612      <1>      loc_next_sum:
1613      <1>      ;xor     ah, ah
1614      <1>      ;test    al, 1
1615      <1>      ;jz     short pass_ah_80h
1616      <1>      ;mov     ah, 80h
1617      <1>      ;pass_ah_80h:
1618      <1>      ;shr     al, 1
1619      0000AA45 D0C8      <1>      ror     al, 1 ; 17/10/2009
1620      0000AA47 0206      <1>      add     al, [esi]
1621      0000AA49 46        <1>      inc     esi
1622      <1>      ;add     al, ah
1623      0000AA4A E2F9      <1>      loop    loc_next_sum
1624      0000AA4C C3        <1>      retn
1625      <1>
1626      <1>      get_fs_longname:
1627      <1>      ; temporary (13/02/2016)
1628      0000AA4D 31C0      <1>      xor     eax, eax
1629      0000AA4F F9        <1>      stc
1630      0000AA50 C3        <1>      retn
1631      <1>
1632      <1>      make_sub_directory:
1633      <1>      ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
1634      <1>      ; 07/08/2022
1635      <1>      ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
1636      <1>      ; 16/10/2016
1637      <1>      ; 02/03/2016, 03/03/2016
1638      <1>      ; 26/02/2016, 27/02/2016
1639      <1>      ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
1640      <1>      ; 01/08/2011 (DIR.ASM, 'proc_make_directory')
1641      <1>      ; 10/07/2010
1642      <1>      ; INPUT ->
1643      <1>      ;     ESI = ASCIIZ Directory Name
1644      <1>      ;     CL = Directory Attributes
1645      <1>      ; OUTPUT ->
1646      <1>      ;     EAX = New sub dir's first cluster
1647      <1>      ;     ESI = Logical Dos Drv Descr. Table Addr.
1648      <1>      ;     CF = 1 -> error code in AL (EAX)
1649      <1>
1650      <1>      ;test    cl, 10h ; directory
1651      <1>      ;jz     short loc_make_directory_access_denied
1652      <1>      ;test    cl, 08h ; volume name
1653      <1>      ;jnz    short loc_make_directory_access_denied
1654      <1>
1655      0000AA51 80E107      <1>      and     cl, 07h
1656      0000AA54 880D[E0800100] <1>      byte [mkdir_attr], cl
1657      <1>
1658      0000AA5A 56        <1>      push    esi
1659      0000AA5B 31DB      <1>      xor     ebx, ebx
1660      0000AA5D 8A3D[62770100] <1>      mov     bh, [Current_Drv]
1661      0000AA63 BE00010900 <1>      mov     esi, Logical_DOSDisks
1662      0000AA68 01DE      <1>      add     esi, ebx
1663      0000AA6A 5B        <1>      pop     ebx
1664      <1>
1665      <1>      ; 10/07/2010 -> 1st writable disk check for trdos
1666      <1>      ; LD_DiskType = 0 for write protection (read only)
1667      0000AA6B 807E0101 <1>      cmp     byte [esi+LD_DiskType], 1 ; 0 = Invalid
1668      0000AA6F 7308      <1>      jnb     short loc_mkdir_check_file_sytem
1669      <1>      ; 16/10/2016 (13h -> 30)
1670      <1>      ;mov     eax, 30 ; 'Disk write-protected' error
1671      <1>      ;mov     edx, 0
1672      <1>      ; 29/07/2022
1673      0000AA71 29C0      <1>      sub     eax, eax
1674      0000AA73 B01E      <1>      mov     al, 30
1675      0000AA75 29D2      <1>      sub     edx, edx ; 0
1676      0000AA77 F9        <1>      stc
1677      <1>      ; err retn: EDX = 0, EBX = Dir name offset
1678      <1>      ;ESI = Logical DOS drive description table address
1679      0000AA78 C3        <1>      retn
1680      <1>
1681      <1>      ;loc_make_directory_access_denied:
1682      <1>      ;mov     ax, 05h ; access denied (invalid attributes input)
1683      <1>      ;stc
1684      <1>      ;retn
1685      <1>
1686      <1>      loc_mkdir_check_file_sytem:
1687      <1>      ; 19/12/2025
1688      <1>      %if 0
1689      <1>      cmp     byte [esi+LD_FATType], 1
1690      <1>      jnb     short loc_mkdir_check_free_sectors
1691      <1>
1692      <1>      loc_make_fs_directory:
1693      <1>      mov     eax, [Current_Dir_FCluster]
1694      <1>
1695      <1>      ; EAX = Parent directory DDT Address
1696      <1>      ; ESI = Logical DOS Drive DT Address
1697      <1>      ; EBX = Directory name offset (as ASCIIZ name)
1698      <1>
1699      <1>      ;call    make_fs_directory
1700      <1>      ;retn
1701      <1>      ; 29/07/2022
1702      <1>      jmp     make_fs_directory
1703      <1>      %endif
1704      <1>
1705      <1>      loc_mkdir_check_free_sectors:
1706      0000AA79 0FB64613 <1>      movzx   eax, byte [esi+LD_BPB+SecPerClust]
1707      0000AA7D 8B4E74      <1>      mov     ecx, [esi+LD_FreeSectors]
1708      0000AA80 39C1      <1>      cmp     ecx, eax
1709      0000AA82 7254      <1>      jb     short loc_mkdir_insufficient_disk_space
1710      <1>
1711      <1>      loc_make_fat_directory:
1712      0000AA84 891D[D0800100] <1>      mov     [mkdir_DirName_Offset], ebx
1713      0000AA8A 890D[DC800100] <1>      mov     [mkdir_FreeSectors], ecx
1714      <1>
1715      <1>      ;mov     al, [esi+LD_BPB+SecPerClust]
1716      0000AA90 A2[E2800100] <1>      mov     byte [mkdir_SecPerClust], al
1717      <1>
1718      <1>      loc_mkdir_gffc_1:

```

```

1719 0000AA95 E8F1160000 <1> call get_first_free_cluster
1720 0000AA9A 722A <1> jc short loc_mkdir_gffc_retn
1721 <1>
1722 <1> ;loc_mkdir_gffc_1_cont:
1723 <1> ;cmp eax, 2
1724 <1> ;jb short loc_mkdir_gffc_insufficient_disk_space
1725 <1>
1726 <1> ;loc_mkdir_gffc_1_save_fcluster:
1727 0000AA9C A3[D4800100] <1> mov [mkdir_FFcluster], eax
1728 <1>
1729 <1> loc_mkdir_locate_ffc:
1730 <1> ; Current directory fcluster <> Directory buffer cluster
1731 <1> ; Current directory will be reloaded by
1732 <1> ; 'locate_current_dir_file' procedure
1733 <1> ;
1734 <1> ; ESI = Logical DOS Drive Description Table Address
1735 <1> ;push esi ; 27/02/2016
1736 0000AAA1 31C0 <1> xor eax, eax
1737 0000AAA3 89C1 <1> mov ecx, eax
1738 0000AAA5 6649 <1> dec cx ; FFFFh
1739 <1> ; CX = FFFFh -> find first deleted or free entry
1740 <1> ; ESI would be ASCIIZ filename address if the call
1741 <1> ; would not be for first free or deleted dir entry
1742 0000AAA7 E84CFBFFFF <1> call locate_current_dir_file
1743 0000AAAC 734A <1> jnc short loc_mkdir_set_ff_dir_entry_1
1744 <1> ;pop esi
1745 <1> ; ESI = Logical DOS Drive Description Table Address
1746 0000AAAE 83F802 <1> cmp eax, 2 ; cmp al, 2 ; File/Dir not found !
1747 0000AAB1 7529 <1> jne short loc_mkdir_stc_return
1748 <1>
1749 <1> loc_mkdir_add_new_cluster:
1750 0000AAB3 3805[61770100] <1> cmp byte [Current_FATtype], al ; 2
1751 <1> ;cmp byte ptr [esi+LD_FATtype], 2
1752 0000AAB9 770C <1> ja short loc_mkdir_add_new_cluster_check_fsc
1753 0000AABB 803D[60770100]01 <1> cmp byte [Current_Dir_Level], 1
1754 <1> ;cmp byte [esi+LD_CDirLevel], 1
1755 0000AAC2 7303 <1> jnb short loc_mkdir_add_new_cluster_check_fsc
1756 <1>
1757 0000AAC4 B00C <1> mov al, 12 ; No more files
1758 <1> loc_mkdir_gffc_retn:
1759 0000AAC6 C3 <1> ret
1760 <1>
1761 <1> loc_mkdir_add_new_cluster_check_fsc:
1762 0000AAC7 8B0D[DC800100] <1> mov ecx, [mkdir_FreeSectors]
1763 <1> ;movzx eax, byte [mkdir_SecPerClust]
1764 0000AACD A0[E2800100] <1> mov al, [mkdir_SecPerClust]
1765 <1> ;shl ax, 1 ; AX = 2 * AX
1766 <1> ; 29/07/2022
1767 0000AAD2 D1E0 <1> shl eax, 1
1768 0000AAD4 39C1 <1> cmp ecx, eax
1769 0000AAD6 7350 <1> jnb short loc_mkdir_add_new_subdir_cluster
1770 <1>
1771 <1> loc_mkdir_insufficient_disk_space:
1772 <1> ;mov edx, ecx
1773 <1> ;loc_mkdir_gffc_insufficient_disk_space:
1774 <1> ; 29/07/2022
1775 <1> ;mov ax, 27h ; MSDOS err => insufficient disk space
1776 <1>
1777 <1> ; err retn: EDX = Free sectors, EBX = Dir name offset
1778 <1> ; ESI -> Dos drive description table address
1779 <1> ; ecx = edx
1780 <1>
1781 <1> ;retn
1782 <1>
1783 <1> ; 29/07/2022
1784 0000AAD8 30E4 <1> xor ah, ah
1785 0000AADA B027 <1> mov al, 27h
1786 <1>
1787 <1> loc_mkdir_stc_return:
1788 0000AADC F9 <1> stc
1789 0000AADD C3 <1> ret
1790 <1>
1791 <1> loc_mkdir_gffc_2:
1792 0000AADE E8A8160000 <1> call get_first_free_cluster
1793 0000AAE3 72E1 <1> jc short loc_mkdir_gffc_retn
1794 <1>
1795 <1> ;loc_mkdir_gffc_1_cont:
1796 <1> ;cmp eax, 2
1797 <1> ;jb short loc_mkdir_gffc_insufficient_disk_space
1798 <1>
1799 <1> ;loc_mkdir_gffc_2_save_fcluster:
1800 0000AAE5 A3[D4800100] <1> mov [mkdir_FFcluster], eax
1801 <1>
1802 0000AAEA A1[D8800100] <1> mov eax, [mkdir_LastDirCluster]
1803 <1>
1804 0000AAEF E841160000 <1> call load_FAT_sub_directory
1805 0000AAF4 72D0 <1> jc short loc_mkdir_gffc_retn
1806 <1>
1807 0000AAF6 31FF <1> xor edi, edi
1808 <1> loc_mkdir_set_ff_dir_entry_1:
1809 <1> ; 27/02/2016
1810 0000AAF8 56 <1> push esi ; Logical DOS Drv Desc. Tbl. address
1811 <1> ; EDI = Directory Entry Address
1812 0000AAF9 8B35[D0800100] <1> mov esi, [mkdir_DirName_Offset]
1813 0000AAFF A1[D4800100] <1> mov eax, [mkdir_FFcluster]
1814 <1>
1815 0000AB04 66B91000 <1> mov cx, 10h ; CL = Directory attribute
1816 <1> ; CH = 0 -> File size is 0
1817 0000AB08 0A0D[E0800100] <1> or cl, [mkdir_attrib] ; S, H, R
1818 0000AB0E E89B010000 <1> call make_directory_entry
1819 <1>
1820 0000AB13 5E <1> pop esi
1821 <1>
1822 0000AB14 C605[897E0100]02 <1> mov byte [DirBuff_ValidData], 2
1823 0000AB1B E861020000 <1> call save_directory_buffer
1824 <1> ;jnc loc_mkdir_set_ff_dir_entry_2
1825 <1> ; 29/07/2022
1826 0000AB20 7205 <1> jc short loc_mkdir_return
1827 0000AB22 E9CA000000 <1> jmp loc_mkdir_set_ff_dir_entry_2
1828 <1>
1829 <1> loc_mkdir_return:
1830 0000AB27 C3 <1> ret
1831 <1>
1832 <1> loc_mkdir_add_new_subdir_cluster:
1833 0000AB28 8B15[8E7E0100] <1> mov edx, [DirBuff_Cluster]
1834 0000AB2E 8915[D8800100] <1> mov [mkdir_LastDirCluster], edx
1835 <1>
1836 0000AB34 A1[D4800100] <1> mov eax, [mkdir_FFcluster]
1837 0000AB39 E8F7150000 <1> call load_FAT_sub_directory
1838 0000AB3E 72E7 <1> jc short loc_mkdir_return
1839 <1> ; eax = 0
1840 <1> ; ecx = directory buffer sector count (<= 128)
1841 <1>
1842 <1> pass_mkdir_add_new_subdir_cluster:

```

```

1843 <1>
1844 <1> ; 29/07/2022
1845 <1> ; sub edi, edi ; 0
1846 <1> ; 29/07/2022 - BUGFIX !
1847 <1> ; mov edi, Directory_Buffer
1848 <1>
1849 <1> ; mov al, 128 ; double word
1850 <1> ; mul ecx ; ecx = directory buffer sector count
1851 <1> ; mov ecx, eax
1852 <1> ; shl cx, 7 ; 128 * sector count
1853 <1> ; mov ax, [esi+LD_BPB+BytesPerSec] ; 512
1854 <1> ; shr ax, 2 ; 'byte count / 4' for 'stosd'
1855 <1> ; 29/07/2022
1856 <1> ; shr eax, 2
1857 <1> ; mul cx ; max = 128*(512/4) -> 16384 (stosd)
1858 <1> ; mov cx, ax
1859 <1> ; sub ax, ax ; 0
1860 <1> ; 29/07/2022
1861 <1> ; mul ecx
1862 <1> ; mov ecx, eax
1863 <1> ; sub eax, eax
1864 <1> ; rep stosd ; clear directory buffer
1865 <1>
1866 <1> ; 29/07/2022
1867 0000AB40 E85C010000 <1> call clear_directory_buffer
1868 <1>
1869 0000AB45 C605[897E0100]02 <1> mov byte [DirBuff_ValidData], 2
1870 0000AB4C E830020000 <1> call save_directory_buffer
1871 0000AB51 72D4 <1> jc short loc_mkdir_return
1872 <1>
1873 <1> loc_mkdir_save_added_cluster:
1874 0000AB53 A1[D8800100] <1> mov eax, [mkdir_LastDirCluster]
1875 0000AB58 8B0D[D4800100] <1> mov ecx, [mkdir_FFCluster]
1876 <1> ; 01/03/2016
1877 0000AB5E 31D2 <1> xor edx, edx
1878 0000AB60 8915[7E7E0100] <1> mov [FAT_ClusterCounter], edx ; 0 ; reset
1879 0000AB66 E8F5160000 <1> call update_cluster
1880 0000AB68 7304 <1> jnc short loc_mkdir_save_fat_buffer_0
1881 0000AB6D 09C0 <1> or eax, eax ; EAX = 0 -> cluster value is 0 or eocc
1882 0000AB6F 7518 <1> jnz short loc_mkdir_save_fat_buffer_stc_retn
1883 <1>
1884 <1> loc_mkdir_save_fat_buffer_0:
1885 0000AB71 A1[D4800100] <1> mov eax, [mkdir_FFCluster]
1886 0000AB76 A3[D8800100] <1> mov [mkdir_LastDirCluster], eax
1887 <1>
1888 0000AB7B 31C9 <1> xor ecx, ecx
1889 0000AB7D 49 <1> dec ecx ; FFFFFFFFh
1890 <1> ; ESI = Logical DOS Drive Description Table address
1891 0000AB7E E8DD160000 <1> call update_cluster
1892 0000AB83 731A <1> jnc short loc_mkdir_save_fat_buffer_1
1893 0000AB85 09C0 <1> or eax, eax
1894 0000AB87 7416 <1> jz short loc_mkdir_save_fat_buffer_1
1895 <1>
1896 <1> loc_mkdir_save_fat_buffer_stc_retn:
1897 <1> ; 01/03/2016
1898 0000AB89 803D[7E7E0100]01 <1> cmp byte [FAT_ClusterCounter], 1
1899 0000AB90 720C <1> jb short loc_mkdir_save_fat_buffer_retn
1900 <1>
1901 0000AB92 66BB00FF <1> mov bx, 0FF00h ; recalculate free space (BL = 0)
1902 <1> ; (BH = FFh -> Use ESI as Drv Param. Tbl.)
1903 0000AB96 50 <1> push eax
1904 0000AB97 E8C5190000 <1> call calculate_fat_freespace
1905 0000AB9C 58 <1> pop eax
1906 0000AB9D F9 <1> stc
1907 <1> loc_mkdir_save_fat_buffer_retn:
1908 0000AB9E C3 <1> retn
1909 <1>
1910 <1> loc_mkdir_save_fat_buffer_1:
1911 <1> ; byte [FAT_BuffValidData] = 2
1912 0000AB9F E830190000 <1> call save_fat_buffer
1913 0000ABA4 72E3 <1> jc short loc_mkdir_save_fat_buffer_stc_retn
1914 <1>
1915 <1> ; 01/03/2016
1916 0000ABA6 803D[7E7E0100]01 <1> cmp byte [FAT_ClusterCounter], 1
1917 0000ABAD 721B <1> jb short loc_mkdir_save_fat_buffer_2
1918 <1>
1919 <1> ; ESI = Logical DOS Drive Description Table address
1920 0000ABAF A1[7E7E0100] <1> mov eax, [FAT_ClusterCounter]
1921 0000ABB4 66BB01FF <1> mov bx, 0FF01h ; add free clusters
1922 0000ABB8 E8A4190000 <1> call calculate_fat_freespace
1923 <1>
1924 <1> ; inc eax ; 0FFFFFFFh -> 0 ; recalculation is needed!
1925 <1> ; jnz short loc_mkdir_save_fat_buffer_2
1926 <1>
1927 <1> ; ecx > 0 -> Recalculation is needed
1928 0000ABBD 09C9 <1> or ecx, ecx
1929 0000ABBF 7409 <1> jz short loc_mkdir_save_fat_buffer_2
1930 <1>
1931 0000ABC1 66BB00FF <1> mov bx, 0FF00h ; ; recalculate free space
1932 0000ABC5 E897190000 <1> call calculate_fat_freespace
1933 <1>
1934 <1> loc_mkdir_save_fat_buffer_2:
1935 0000ABCA C605[E3800100]01 <1> mov byte [mkdir_add_new_cluster], 1
1936 0000ABD1 E9B0000000 <1> jmp loc_mkdir_upd_parent_dir_lmdt
1937 <1>
1938 <1> loc_mkdir_update_sub_dir_cluster:
1939 0000ABD6 A1[D4800100] <1> mov eax, [mkdir_FFCluster]
1940 0000ABDB 29C9 <1> sub ecx, ecx ; 0
1941 <1> ; 01/03/2016
1942 0000ABDD 890D[7E7E0100] <1> mov [FAT_ClusterCounter], ecx ; 0 ; Reset
1943 0000ABE3 49 <1> dec ecx ; 0FFFFFFFh
1944 <1>
1945 <1> ; ESI = Logical DOS Drive Description Table address
1946 0000ABE4 E877160000 <1> call update_cluster
1947 0000ABE9 7364 <1> jnc short loc_mkdir_save_fat_buffer_3
1948 0000ABEB 09C0 <1> or eax, eax ; EAX = 0 -> cluster value is 0 or eocc
1949 0000ABED 7460 <1> jz short loc_mkdir_save_fat_buffer_3
1950 <1> ; 01/03/2016
1951 0000ABEF EB98 <1> jmp short loc_mkdir_save_fat_buffer_stc_retn
1952 <1>
1953 <1> loc_mkdir_set_ff_dir_entry_2:
1954 <1> ; ESI = Logical DOS Drive Description Table address
1955 0000ABF1 A1[D4800100] <1> mov eax, [mkdir_FFCluster]
1956 <1> ; Load disk sectors as a directory cluster
1957 0000ABF6 E83A150000 <1> call load_FAT_sub_directory
1958 0000ABFB 7251 <1> jc short retn_make_fat_directory
1959 <1>
1960 <1> ; eax = 0
1961 <1> ; ecx = directory buffer sector count (<= 128)
1962 <1>
1963 <1> ; 29/07/2022
1964 <1> ; mov edi, Directory_Buffer + 64 ; 26/02/2016
1965 <1>
1966 <1> ; 02/03/2016

```

```

1967 <1> ; mov ax, [esi+LD_BPB+BytesPerSec] ; 512
1968 <1> ; shr ax, 2 ; 'byte count / 4' for 'stosd'
1969 <1> ; ; 29/07/2022
1970 <1> ; shr eax, 2
1971 <1> ; mul ecx
1972 <1> ; mov ecx, eax
1973 <1> ; ;
1974 <1> ; ; 29/07/2022 - BUGFIX !
1975 <1> ; sub ecx, 16 ; - 64 bytes
1976 <1> ; ; (space for '.' & '..' entries)
1977 <1> ; sub ax, ax
1978 <1> ; sub eax, eax
1979 <1> ; rep stosd
1980 <1> ; ;
1981 <1> ; ; mov al, 128 ; double word (count in sector)
1982 <1> ; ; mul ecx ; ecx = directory buffer sector count
1983 <1> ; ; mov ecx, eax
1984 <1> ; ; shl cx, 7 ; 128 * sector count
1985 <1> ; ; sub ecx, 64 ; 29/07/2022
1986 <1> ; ; sub eax, eax
1987 <1> ; ; sub al, al ; 0
1988 <1> ; ; rep stosd ; clear directory buffer
1989 <1> ;
1990 <1> ; ; 29/07/2022
1991 0000ABFD E89F000000 <1> call clear_directory_buffer
1992 <1> ;
1993 0000AC02 BF00000800 <1> mov edi, Directory_Buffer ; 26/02/2016
1994 <1> ;
1995 0000AC07 56 <1> push esi
1996 <1> ;
1997 0000AC08 BE[E4800100] <1> mov esi, mkdir_Name
1998 0000AC0D 66C7062E00 <1> mov word [esi], 2Eh ; db '.', '0'
1999 <1> ;
2000 0000AC12 A1[D4800100] <1> mov eax, [mkdir_FFcluster]
2001 <1> ; mov cx, 10h ; CL = Directory attribute
2002 <1> ; ; CH = 0 -> File size is 0
2003 <1> ; ; 29/07/2022
2004 0000AC17 B110 <1> mov cl, 10h
2005 0000AC19 E890000000 <1> call make_directory_entry
2006 <1> ;
2007 0000AC1E BF20000800 <1> mov edi, Directory_Buffer + 32 ; 26/02/2016
2008 <1> ;
2009 <1> ; ; 03/03/2016
2010 <1> ; ; Following modification has been done according to
2011 <1> ; ; 'Microsoft Extensible Firmware Initiative
2012 <1> ; ; FAT32 File System Specification' document,
2013 <1> ; ; 'FAT: General Overview of On-Disk Format-Page 25'.
2014 <1> ; ; "Finally, you set DIR_FstClusLO and DIR_FstClusHI
2015 <1> ; ; for the dotdot entry (the second entry) to the
2016 <1> ; ; first cluster number of the directory in which you
2017 <1> ; ; just created the directory (value is 0 if this directory
2018 <1> ; ; is the root directory even for FAT32 volumes)."
2019 <1> ; ; (Correctness of this modification has been verified
2020 <1> ; ; by using windows 98 'scandisk.exe'.)
2021 <1> ;
2022 0000AC23 29C0 <1> sub eax, eax
2023 0000AC25 3805[60770100] <1> cmp byte [Current_Dir_Level], al ; 0
2024 0000AC2B 7605 <1> jna short loc_mkdir_set_ff_dir_entry_3
2025 0000AC2D A1[5C770100] <1> mov eax, [Current_Dir_FCluster] ; parent dir
2026 <1> loc_mkdir_set_ff_dir_entry_3:
2027 0000AC32 66C746012E00 <1> mov word [esi+1], 2Eh ; db '.', '0'
2028 <1> ;
2029 <1> ; mov cx, 10h
2030 0000AC38 E871000000 <1> call make_directory_entry
2031 <1> ;
2032 0000AC3D 5E <1> pop esi
2033 <1> ;
2034 0000AC3E C605[897E0100]02 <1> mov byte [DirBuff_ValidData], 2
2035 0000AC45 E837010000 <1> call save_directory_buffer
2036 <1> ; jnc loc_mkdir_update_sub_dir_cluster
2037 <1> ; ; 29/07/2022
2038 0000AC4A 7202 <1> jc short retn_make_fat_directory
2039 0000AC4C E888 <1> jmp loc_mkdir_update_sub_dir_cluster
2040 <1> ;
2041 <1> retn_make_fat_directory:
2042 0000AC4E C3 <1> retn
2043 <1> ;
2044 <1> loc_mkdir_save_fat_buffer_3:
2045 <1> ; ; 01/03/2016
2046 <1> ; ; byte [FAT_BuffValidData] = 2
2047 0000AC4F E880180000 <1> call save_fat_buffer
2048 <1> ; jc short loc_mkdir_save_fat_buffer_stc_retn
2049 <1> ; ; 07/08/2022
2050 0000AC54 7305 <1> jnc short loc_mkdir_save_fat_buffer_4
2051 0000AC56 E92EFFFFFF <1> jmp loc_mkdir_save_fat_buffer_stc_retn
2052 <1> ;
2053 <1> loc_mkdir_save_fat_buffer_4:
2054 0000AC5B 803D[7E7E0100]01 <1> cmp byte [FAT_ClusterCounter], 1
2055 0000AC62 721B <1> jb short loc_mkdir_save_fat_buffer_5
2056 <1> ;
2057 <1> ; ESI = Logical DOS Drive Description Table address
2058 0000AC64 A1[7E7E0100] <1> mov eax, [FAT_ClusterCounter]
2059 0000AC69 66BB01FF <1> mov bx, 0FF01h ; add free clusters
2060 0000AC6D E8EF180000 <1> call calculate_fat_freespace
2061 <1> ;
2062 <1> ; inc eax ; 0FFFFFFFh -> 0 ; recalculation is needed!
2063 <1> ; jnz short loc_mkdir_save_fat_buffer_5
2064 <1> ;
2065 <1> ; ecx > 0 -> Recalculation is needed
2066 0000AC72 09C9 <1> or ecx, ecx
2067 0000AC74 7409 <1> jz short loc_mkdir_save_fat_buffer_5
2068 <1> ;
2069 0000AC76 66BB00FF <1> mov bx, 0FF00h ; recalculate free space
2070 0000AC7A E8E2180000 <1> call calculate_fat_freespace
2071 <1> ;
2072 <1> loc_mkdir_save_fat_buffer_5:
2073 0000AC7F C605[E3800100]00 <1> mov byte [mkdir_add_new_cluster], 0
2074 <1> ;
2075 <1> loc_mkdir_upd_parent_dir_lmdt:
2076 0000AC86 E87F010000 <1> call update_parent_dir_lmdt
2077 <1> ;
2078 <1> ; ; 01/03/2016
2079 0000AC8B 803D[E3800100]00 <1> cmp byte [mkdir_add_new_cluster], 0
2080 <1> ; ja loc_mkdir_gffc_2
2081 <1> ; ; 29/07/2022
2082 0000AC92 7605 <1> jna short loc_mkdir_retn_new_dir_cluster
2083 0000AC94 E945FFFFFF <1> jmp loc_mkdir_gffc_2
2084 <1> ;
2085 <1> loc_mkdir_retn_new_dir_cluster:
2086 0000AC99 A1[D4800100] <1> mov eax, [mkdir_FFcluster]
2087 0000AC9E 31D2 <1> xor edx, edx
2088 <1> loc_mkdir_retn:
2089 0000ACA0 C3 <1> retn
2090 <1> ;

```



```

2091 <1> clear_directory_buffer:
2092 <1> ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
2093 <1> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
2094 <1> ;
2095 <1> ; eax = 0
2096 <1> ; ecx = directory buffer sector count (<= 128)
2097 <1> ;
2098 0000ACA1 BF00000800 <1> mov edi, Directory_Buffer
2099 <1> ;mov al, 128 ; double word
2100 <1> ;mul ecx ; ecx = directory buffer sector count
2101 <1> ;mov ecx, eax
2102 <1> ;shl cx, 7 ; 128 * sector count
2103 <1> ; 19/12/2025
2104 <1> ;mov ax, [esi+LD_BPB+BytesPerSec] ; 512
2105 <1> ;shr eax, 2 ; 'byte count / 4' for 'stosd'
2106 <1> ;mov ax, 128 ; 512/4
2107 <1> ;mul ecx ; max = 128*(512/4) -> 16384 (stosd)
2108 <1> ;mov ecx, eax
2109 0000ACA6 C1E107 <1> shl ecx, 7 ; (512/4) * sector count
2110 0000ACA9 29C0 <1> sub eax, eax
2111 0000ACAB F3AB <1> rep stosd ; clear directory buffer
2112 0000ACAD C3 <1> retn
2113 <1>
2114 <1> make_directory_entry:
2115 <1> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
2116 <1> ; 02/03/2016
2117 <1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
2118 <1> ; 09/08/2010 (DIR.ASM, 'proc_make_directory_entry')
2119 <1> ; 17/07/2010
2120 <1> ; INPUT ->
2121 <1> ; EDI = Directory Entry Address
2122 <1> ; ESI = Dot File Name Location
2123 <1> ; EAX = First Cluster
2124 <1> ; File Size = 0 (Must be set later)
2125 <1> ; CL = Attributes
2126 <1> ; CH = 0 (File size = 0)
2127 <1> ; (If CH>0, File size is in dword [EBX]) (*)
2128 <1> ; OUTPUT ->
2129 <1> ; EDI = Directory Entry Address
2130 <1> ; ESI = Dot File Name Location (Capitalized)
2131 <1> ; If CH input = 0, File Size = 0
2132 <1> ; Otherwise file size is as dword [EBX] (*)
2133 <1> ; DX = Date, AX = Time in DOS Dir Entry format
2134 <1> ; EBX = same
2135 <1> ; ECX = same
2136 <1>
2137 0000ACAE 51 <1> push ecx
2138 <1>
2139 0000ACAF 884F0B <1> mov [edi+11], cl ; Attributes
2140 0000ACB2 6689471A <1> mov [edi+26], ax ; FClusterLw, 26
2141 0000ACB6 C1E810 <1> shr eax, 16
2142 0000ACB9 66894714 <1> mov [edi+20], ax ; FClusterHw, 20
2143 <1> ;xor ax, ax
2144 <1> ; 29/07/2022
2145 0000ACBD 31C0 <1> xor eax, eax
2146 0000ACBF 6689470C <1> mov [edi+12], ax ; NTReserved, 12
2147 <1> ; CrtTimeTenth, 13
2148 0000ACC3 08ED <1> or ch, ch
2149 0000ACC5 7402 <1> jz short loc_make_direntry_set_filesize
2150 <1>
2151 0000ACC7 8B03 <1> mov eax, [ebx]
2152 <1>
2153 <1> loc_make_direntry_set_filesize:
2154 0000ACC9 89471C <1> mov [edi+28], eax ; FileSize, 28
2155 <1>
2156 0000ACCC E8D3FBFFFF <1> call convert_file_name
2157 <1> ; EDI = Dir Entry Format File Name Location
2158 <1> ; ESI = Dot File Name Location (capitalized)
2159 <1>
2160 0000ACD1 E816000000 <1> call convert_current_date_time
2161 <1> ; OUTPUT -> DX = Date in dos dir entry format
2162 <1> ; AX = Time in dos dir entry format
2163 0000ACD6 6689470E <1> mov [edi+14], ax ; CrtTime, 14
2164 0000ACDA 66895710 <1> mov [edi+16], dx ; CrtDate, 16
2165 0000ACDE 66895712 <1> mov [edi+18], dx ; LastAccDate, 18
2166 0000ACE2 66894716 <1> mov [edi+22], ax ; WrtTime, 14
2167 0000ACE6 66895718 <1> mov [edi+24], dx ; WrtDate, 16
2168 0000ACEA 59 <1> pop ecx
2169 <1>
2170 0000ACEB C3 <1> retn
2171 <1>
2172 <1> convert_current_date_time:
2173 <1> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
2174 <1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
2175 <1> ; 13/06/2010 (DIR.ASM, 'proc_convert_current_date_time')
2176 <1> ; converts date&time to dos dir entry format
2177 <1> ; INPUT -> none
2178 <1> ; OUTPUT -> DX = Date in dos dir entry format
2179 <1> ; AX = Time in dos dir entry format
2180 <1>
2181 0000ACEC B404 <1> mov ah, 04h ; Return Current Date
2182 0000ACEE E8EAB5FFFF <1> call int1Ah
2183 <1>
2184 0000ACF3 88E8 <1> mov al, ch ; <- century BCD
2185 0000ACF5 240F <1> and al, 0Fh
2186 0000ACF7 88EC <1> mov ah, ch
2187 0000ACF9 C0EC04 <1> shr ah, 4
2188 0000ACFC D50A <1> aad
2189 0000ACFE 88C5 <1> mov ch, al ; -> century
2190 <1>
2191 0000AD00 88C8 <1> mov al, cl ; <- year BCD
2192 0000AD02 240F <1> and al, 0Fh
2193 0000AD04 88CC <1> mov ah, cl
2194 0000AD06 C0EC04 <1> shr ah, 4
2195 0000AD09 D50A <1> aad
2196 0000AD0B 88C1 <1> mov cl, al ; -> year
2197 <1>
2198 <1> ;mov al, ch
2199 <1> ;mov ah, 100
2200 <1> ;mul ah
2201 <1> ; 29/07/2022
2202 0000AD0D B064 <1> mov al, 100
2203 0000AD0F F6E5 <1> mul ch
2204 <1>
2205 <1> ;xor ch, ch
2206 <1> ;add ax, cx
2207 <1> ; 29/07/2022
2208 0000AD11 00C8 <1> add al, cl
2209 0000AD13 80D400 <1> adc ah, 0
2210 0000AD16 662DBC07 <1> sub ax, 1980 ; ms-dos epoch
2211 <1> ;mov cx, ax
2212 <1> ; 29/07/2022
2213 0000AD1A 88C1 <1> mov cl, al
2214 <1> ;mov ecx, eax

```

```

2215 <1>
2216 0000AD1C 88F0 <1> mov al, dh ; <- month in bcd
2217 0000AD1E 240F <1> and al, 0Fh
2218 0000AD20 88F4 <1> mov ah, dh
2219 0000AD22 C0EC04 <1> shr ah, 4
2220 0000AD25 D50A <1> aad
2221 0000AD27 88C6 <1> mov dh, al ; -> month
2222 <1>
2223 0000AD29 88D0 <1> mov al, dl ; <- day BCD
2224 0000AD2B 240F <1> and al, 0Fh
2225 0000AD2D 88D4 <1> mov ah, dl
2226 0000AD2F C0EC04 <1> shr ah, 4
2227 0000AD32 D50A <1> aad
2228 0000AD34 88C2 <1> mov dl, al ; -> day
2229 <1>
2230 0000AD36 88C8 <1> mov al, cl ; count of years from 1980
2231 <1> ;shl ax, 4
2232 <1> ; 29/07/2022
2233 <1> ;mov eax, ecx
2234 0000AD38 C1E004 <1> shl eax, 4
2235 <1>
2236 0000AD3B 08F0 <1> or al, dh ; month of year, 1 to 12
2237 <1> ;shl ax, 5
2238 <1> ; 29/07/2022
2239 0000AD3D C1E005 <1> shl eax, 5
2240 0000AD40 08D0 <1> or al, dl ; day of year, 1 to 31
2241 <1>
2242 <1> ;push ax ; push date
2243 <1> ; 29/07/2022
2244 0000AD42 50 <1> push eax
2245 <1>
2246 0000AD43 B402 <1> mov ah, 02h ; Return Current Time
2247 0000AD45 E893B5FFFF <1> call int1Ah
2248 <1>
2249 0000AD4A 88E8 <1> mov al, ch ; <- hours BCD
2250 0000AD4C 240F <1> and al, 0Fh
2251 0000AD4E 88EC <1> mov ah, ch
2252 0000AD50 C0EC04 <1> shr ah, 4
2253 0000AD53 D50A <1> aad
2254 0000AD55 88C5 <1> mov ch, al ; -> hours
2255 <1>
2256 0000AD57 88C8 <1> mov al, cl ; <- minutes BCD
2257 0000AD59 240F <1> and al, 0Fh
2258 0000AD5B 88CC <1> mov ah, cl
2259 0000AD5D C0EC04 <1> shr ah, 4
2260 0000AD60 D50A <1> aad
2261 0000AD62 88C1 <1> mov cl, al ; -> minutes
2262 <1>
2263 0000AD64 88F0 <1> mov al, dh ; <- seconds BCD
2264 0000AD66 240F <1> and al, 0Fh
2265 0000AD68 88F4 <1> mov ah, dh
2266 0000AD6A C0EC04 <1> shr ah, 4
2267 0000AD6D D50A <1> aad
2268 0000AD6F 88C6 <1> mov dh, al ; -> seconds
2269 <1>
2270 0000AD71 88E8 <1> mov al, ch ; hours
2271 <1> ;shl ax, 6
2272 <1> ; 29/07/2022
2273 0000AD73 C1E006 <1> shl eax, 6
2274 0000AD76 08C8 <1> or al, cl ; minutes
2275 <1> ;shl ax, 5
2276 <1> ; 29/07/2022
2277 0000AD78 C1E005 <1> shl eax, 5
2278 0000AD7B D0EE <1> shr dh, 1 ; 2 seconds
2279 <1> ; There is a bug in TRDOS v1 here !
2280 <1> ; it was 'or al, dl' !
2281 0000AD7D 08F0 <1> or al, dh ; seconds
2282 <1>
2283 <1> ;pop dx ; pop date
2284 <1> ; 29/07/2022
2285 0000AD7F 5A <1> pop edx
2286 <1>
2287 0000AD80 C3 <1> retn
2288 <1>
2289 <1> save_directory_buffer:
2290 <1> ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
2291 <1> ; 30/07/2022
2292 <1> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
2293 <1> ; 15/10/2016
2294 <1> ; 23/03/2016
2295 <1> ; 26/02/2016
2296 <1> ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
2297 <1> ; 01/08/2011
2298 <1> ; 14/03/2010
2299 <1> ; INPUT ->
2300 <1> ; none
2301 <1> ; OUTPUT ->
2302 <1> ; cf = 0 -> write OK...
2303 <1> ; cf = 1 -> error code in AL (EAX)
2304 <1> ; cf = 1 & AL = 0Dh => CH & CL = FS & FAT type
2305 <1> ; EBX = Directory Buffer Address
2306 <1> ;
2307 <1> ; (EAX, ECX, EDX will be modified)
2308 <1>
2309 0000AD81 BB00000800 <1> mov ebx, Directory_Buffer
2310 0000AD86 803D[897E0100]02 <1> cmp byte [DirBuff_ValidData], 2
2311 0000AD8D 7403 <1> je short loc_save_dir_buffer
2312 0000AD8F 31C0 <1> xor eax, eax
2313 0000AD91 C3 <1> retn
2314 <1>
2315 <1> loc_save_dir_buffer:
2316 0000AD92 56 <1> push esi
2317 0000AD93 31DB <1> xor ebx, ebx
2318 0000AD95 8A3D[877E0100] <1> mov bh, [DirBuff_DRV]
2319 0000AD9B 80EF41 <1> sub bh, 'A'
2320 0000AD9E BE00010900 <1> mov esi, Logical_DOSDisks
2321 0000ADA3 01DE <1> add esi, ebx
2322 <1>
2323 <1> ; 19/12/2025
2324 <1> %if 0
2325 <1> mov cx, [esi+LD_FATType]
2326 <1> ; CH = FS Type (A1h for FS)
2327 <1> ; CL = FAT Type (0 for FS)
2328 <1> or cl, cl
2329 <1> jz short loc_save_dir_buff_stc_retn
2330 <1> %endif
2331 <1>
2332 <1> loc_save_dir_buffer_check_cluster_no:
2333 0000ADA5 A1[8E7E0100] <1> mov eax, [DirBuff_Cluster]
2334 0000ADAA 28FF <1> sub bh, bh ; ebx = 0
2335 0000ADAC 09C0 <1> or eax, eax
2336 0000ADAE 7537 <1> jnz short loc_save_sub_dir_buffer
2337 0000ADB0 8A25[887E0100] <1> mov ah, [DirBuff_FATType]
2338 <1>

```

```

2339 <1> ; 19/12/2025
2340 <1> %if 0
2341 <1> inc b1 ; b1 = 1
2342 <1> cmp ah, b1
2343 <1> jb short loc_save_dir_buff_inv_data_retn
2344 <1> inc b1 ; b1 = 2
2345 <1> cmp b1, ah
2346 <1> jb short loc_save_dir_buff_inv_data_retn
2347 <1> %else
2348 <1> ; 19/12/2025
2349 0000ADB6 80FC03 <1> cmp ah, 3 ; FAT32 ?
2350 0000ADB9 7314 <1> jnb short loc_save_dir_buff_stc_retn ; yes
2351 <1> %endif
2352 <1>
2353 <1> loc_save_root_dir_buffer:
2354 0000ADBB 668B5E17 <1> mov bx, [esi+LD_BPB+RootDirEnts]
2355 0000ADBF 6683C30F <1> add bx, 15
2356 <1> ;shr bx, 4 ; 16 dir entries per sector
2357 <1> ; 29/07/2022
2358 0000ADC3 C1EB04 <1> shr ebx, 4
2359 <1> ;or bx, bx
2360 0000ADC6 09DB <1> or ebx, ebx
2361 0000ADC8 7405 <1> jz short loc_save_dir_buff_stc_retn
2362 <1> ;mov ecx, ebx
2363 0000ADCA 8B4664 <1> mov eax, [esi+LD_ROOTBegin] ; 26/02/2016
2364 0000ADCD EB22 <1> jmp short loc_write_directory_to_disk
2365 <1>
2366 <1> loc_save_dir_buff_stc_retn:
2367 0000ADCF F9 <1> stc
2368 <1> loc_save_dir_buff_inv_data_retn:
2369 <1> ; 15/10/2016 (0Dh -> 29)
2370 0000ADD0 B01D <1> mov al, 29 ; Invalid data !
2371 0000ADD2 C605[897E0100]00 <1> mov byte [DirBuff_ValidData], 0
2372 0000ADD9 EB05 <1> jmp short loc_save_dir_buff_retn
2373 <1>
2374 <1> loc_write_directory_to_disk_err:
2375 <1> ; 15/10/2016 (disk write error code, 1Dh -> 18)
2376 0000ADB8 B812000000 <1> mov eax, 18 ; Drive not ready or write error
2377 <1>
2378 <1> loc_save_dir_buff_retn:
2379 0000ADE0 BB00000800 <1> mov ebx, Directory_Buffer
2380 0000ADE5 5E <1> pop esi
2381 0000ADE6 C3 <1> retn
2382 <1>
2383 <1> loc_save_sub_dir_buffer:
2384 <1> ; ebx = 0
2385 <1> ;sub eax, 2
2386 <1> ; 30/07/2022
2387 0000ADE7 48 <1> dec eax
2388 0000ADE8 48 <1> dec eax
2389 0000ADE9 8A5E13 <1> mov b1, [esi+LD_BPB+SecPerClust]
2390 0000ADEC F7E3 <1> mul ebx
2391 0000ADEE 034668 <1> add eax, [esi+LD_DATABegin]
2392 <1> ;mov ecx, ebx
2393 <1>
2394 <1> loc_write_directory_to_disk:
2395 0000ADF1 89D9 <1> mov ecx, ebx
2396 0000ADF3 BB00000800 <1> mov ebx, Directory_Buffer
2397 0000ADF8 E8D76B0000 <1> call disk_write
2398 0000ADFD 72DC <1> jc short loc_write_directory_to_disk_err
2399 <1>
2400 <1> loc_save_dir_buff_validate_retn:
2401 0000ADFF C605[897E0100]01 <1> mov byte [DirBuff_ValidData], 1
2402 0000AE06 31C0 <1> xor eax, eax
2403 <1> ; 26/02/2016
2404 0000AE08 EBD6 <1> jmp short loc_save_dir_buff_retn
2405 <1>
2406 <1> update_parent_dir_lmdt:
2407 <1> ; 19/12/2025
2408 <1> ; 16/07/2025 (TRDOS 386 kernel v2.0.10) (BugFix)
2409 <1> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
2410 <1> ; 29/12/2017
2411 <1> ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
2412 <1> ; 01/08/2011
2413 <1> ; 16/10/2010
2414 <1> ;
2415 <1> ; INPUT ->
2416 <1> ; none
2417 <1> ; OUTPUT ->
2418 <1> ; (last modification date & time of the parent dir
2419 <1> ; will be changed/updated)
2420 <1> ;
2421 <1> ; (EAX, EBX, ECX, EDX, EDI will be changed)
2422 <1>
2423 0000AE0A 29C0 <1> sub eax, eax
2424 0000AE0C 8A25[60770100] <1> mov ah, [Current_Dir_Level]
2425 <1>
2426 <1> ; 19/12/2025
2427 <1> %if 0
2428 <1> mov al, [Current_FATType]
2429 <1> cmp al, 1
2430 <1> jb short loc_UPDLMDT_proc_retn
2431 <1> %endif
2432 <1>
2433 <1> loc_update_parent_dir_lm_date_time:
2434 0000AE12 08E4 <1> or ah, ah
2435 0000AE14 7433 <1> jz short loc_UPDLMDT_proc_retn
2436 <1>
2437 0000AE16 56 <1> push esi ; *
2438 0000AE17 8825[CB800100] <1> mov [UPDLMDT_CDirLevel], ah
2439 0000AE1D 8B15[5C770100] <1> mov edx, [Current_Dir_FCluster]
2440 0000AE23 8915[CC800100] <1> mov [UPDLMDT_CDirFCluster], edx
2441 <1>
2442 0000AE29 FECC <1> dec ah
2443 <1> ;mov ecx, 12
2444 <1> ; 29/07/2022
2445 0000AE2B 29C9 <1> sub ecx, ecx
2446 0000AE2D B10C <1> mov cl, 12
2447 <1> ;
2448 0000AE2F BE[C07E0100] <1> mov esi, PATH_Array
2449 <1>
2450 0000AE34 8825[60770100] <1> mov [Current_Dir_Level], ah
2451 0000AE3A 08E4 <1> or ah, ah
2452 0000AE3C 750C <1> jnz short loc_update_parent_dir_lmdt_load_sub_dir_1
2453 0000AE3E 803D[61770100]02 <1> cmp byte [Current_FATType], 2
2454 0000AE45 7709 <1> ja short loc_update_parent_dir_lmdt_load_sub_dir_2
2455 <1> ; 19/12/2025
2456 <1> ;sub al, al
2457 <1> ; eax = 0
2458 0000AE47 EB0A <1> jmp short loc_update_parent_dir_lmdt_load_sub_dir_3
2459 <1>
2460 <1> loc_UPDLMDT_proc_retn:
2461 0000AE49 C3 <1> retn
2462 <1>

```

```

2463      <1> loc_update_parent_dir_lmdt_load_sub_dir_1:
2464 0000AE4A B010      <1>     mov     al, 16
2465 0000AE4C F6E4      <1>     mul     ah
2466 0000AE4E 01C6      <1>     add     esi, eax
2467      <1>
2468      <1> loc_update_parent_dir_lmdt_load_sub_dir_2:
2469 0000AE50 8B460C      <1>     mov     eax, [esi+12] ; Parent Dir First Cluster
2470      <1>
2471      <1> loc_update_parent_dir_lmdt_load_sub_dir_3:
2472 0000AE53 A3[5C770100]    <1>     mov     [Current_Dir_FCluster], eax
2473      <1>
2474 0000AE58 83C610      <1>     add     esi, 16
2475      <1>     ;mov     di, Dir_File_Name
2476      <1>     ; 16/07/2025 (BugFix)
2477 0000AE5B BFE67F0100]    <1>     mov     edi, Dir_File_Name
2478 0000AE60 F3A4      <1>     rep     movsb
2479      <1>
2480 0000AE62 BE00010900      <1>     mov     esi, Logical_DOSDisks
2481 0000AE67 29DB      <1>     sub     ebx, ebx
2482 0000AE69 8A3D[62770100]    <1>     mov     bh, [Current_Drv]
2483 0000AE6F 01DE      <1>     add     esi, ebx
2484 0000AE71 E83BF8FFFF      <1>     call    reload_current_directory
2485 0000AE76 722F      <1>     jc      short loc_update_parent_dir_lmdt_restore_cdirlevel
2486      <1>
2487      <1> loc_update_parent_dir_lmdt_locate_dir:
2488 0000AE78 BEE67F0100]    <1>     mov     esi, Dir_File_Name
2489      <1>     ;xor     cx, cx
2490      <1>     ; 29/07/2022
2491 0000AE7D 31C9      <1>     xor     ecx, ecx
2492 0000AE7F 66B81008      <1>     mov     ax, 0810h ; Only directories
2493 0000AE83 E870F7FFFF      <1>     call    locate_current_dir_file
2494      <1>     ; EDI = DirBuff Directory Entry Address
2495 0000AE88 721D      <1>     jc      short loc_update_parent_dir_lmdt_restore_cdirlevel
2496      <1>
2497 0000AE8A E85DFEFFFF      <1>     call    convert_current_date_time
2498 0000AE8F 66895712      <1>     mov     [edi+18], dx ; Last Access Date
2499 0000AE93 66895718      <1>     mov     [edi+24], dx ; Last Write Date
2500 0000AE97 66894716      <1>     mov     [edi+22], ax ; Last Write Time
2501      <1>
2502 0000AE9B C605[897E0100]02    <1>     mov     byte [DirBuff_ValidData], 2
2503 0000AEA2 E8DAFEFFFF      <1>     call    save_directory_buffer
2504      <1>     ; 29/12/2017
2505      <1>     ;jc      short loc_update_parent_dir_lmdt_restore_cdirlevel
2506      <1>     ;xor     al, al
2507      <1> loc_update_parent_dir_lmdt_restore_cdirlevel:
2508      <1>     ;current directory level restoration
2509 0000AEA7 8A25[CB800100]    <1>     mov     ah, [UPDLMDT_CDirLevel]
2510 0000AEAD 8825[60770100]    <1>     mov     [Current_Dir_Level], ah
2511 0000AEB3 8B15[CC800100]    <1>     mov     edx, [UPDLMDT_CDirFCluster]
2512 0000AEB9 8915[5C770100]    <1>     mov     [Current_Dir_FCluster], edx
2513      <1>
2514 0000AEBF 5E      <1>     pop     esi ; *
2515 0000AEC0 C3      <1>     retn
2516      <1>
2517      <1> delete_longname:
2518      <1>     ; 29/07/2022 (TRDOS 386 Kernel v2.0.5)
2519      <1>     ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)
2520      <1>     ; 01/08/2011 (DIR.ASM, 'proc_delete_longname')
2521      <1>     ; 14/03/2010
2522      <1>     ; INPUT ->
2523      <1>     ; EAX = Directory Entry (Index) Number (< 65536)
2524      <1>     ; OUTPUT ->
2525      <1>     ; cf = 0 -> OK (EAX = 0)
2526      <1>     ; cf = 1 -> error code in EAX (AL)
2527      <1>     ;
2528      <1>     ; (Modified registers: EAX, EDX, ECX, EBX, EDI)
2529      <1>
2530 0000AEC1 66A3[34810100]    <1>     mov     [DLN_EntryNumber], ax
2531 0000AEC7 C605[36810100]40    <1>     mov     byte [DLN_40h], 40h
2532      <1>
2533 0000AECE E857000000      <1>     call    locate_current_dir_entry
2534 0000AED3 7307      <1>     jnc     short loc_dln_check_attributes
2535 0000AED5 C3      <1>     retn
2536      <1>
2537      <1> loc_dln_longname_not_found:
2538      <1>     ;mov     eax, 2
2539      <1>     ; 29/07/2022
2540 0000AED6 29C0      <1>     sub     eax, eax
2541 0000AED8 B002      <1>     mov     al, 2
2542 0000AEDA F9      <1>     stc
2543 0000AEDB C3      <1>     retn
2544      <1>
2545      <1> loc_dln_check_attributes:
2546 0000AEDC B00F      <1>     mov     al, 0Fh ; long name
2547 0000AEDE 8A670B      <1>     mov     ah, [edi+0Bh] ; dir entry attributes
2548 0000AEE1 38C4      <1>     cmp     ah, al
2549 0000AEE3 75F1      <1>     jne     short loc_dln_longname_not_found
2550 0000AEE5 8A27      <1>     mov     ah, [edi]
2551 0000AEE7 2A25[36810100]    <1>     sub     ah, [DLN_40h]
2552 0000AEEF 76E7      <1>     jna     short loc_dln_longname_not_found
2553 0000AEF0 80FC14      <1>     cmp     ah, 14h ; 84-64=20 -> 20*13=260 bytes
2554 0000AEF2 77E2      <1>     ja      short loc_dln_longname_not_found
2555      <1>
2556 0000AEF4 C607E5      <1>     mov     byte [edi], 0E5h ; deleted sign
2557 0000AEF7 C605[897E0100]02    <1>     mov     byte [DirBuff_ValidData], 2 ; changed/write sign
2558 0000AEFE C605[36810100]00    <1>     mov     byte [DLN_40h], 0 ; 40h -> 0
2559      <1>
2560      <1> loc_dln_delete_next_ln_entry:
2561 0000AF05 80FC01      <1>     cmp     ah, 1
2562 0000AF08 7616      <1>     jna     short loc_dln_longname_retn
2563      <1>
2564 0000AF0A 66FF05[34810100]    <1>     inc     word [DLN_EntryNumber]
2565 0000AF11 0FB705[34810100]    <1>     movzx   eax, word [DLN_EntryNumber]
2566 0000AF18 E80D000000      <1>     call    locate_current_dir_entry
2567 0000AF1D 73BD      <1>     jnc     short loc_dln_check_attributes
2568      <1>
2569      <1> loc_dln_longname_stc_retn:
2570 0000AF1F C3      <1>     retn
2571      <1>
2572      <1> loc_dln_longname_retn:
2573      <1>     ;cmp     byte [DirBuff_ValidData], 2
2574      <1>     ;jne     short loc_dln_longname_retn_xor_eax
2575 0000AF20 E85CFEFFFF      <1>     call    save_directory_buffer
2576 0000AF25 72F8      <1>     jc      short loc_dln_longname_stc_retn
2577      <1>
2578      <1> loc_dln_longname_retn_xor_eax:
2579 0000AF27 31C0      <1>     xor     eax, eax
2580 0000AF29 C3      <1>     retn
2581      <1>
2582      <1> locate_current_dir_entry:
2583      <1>     ; 19/12/2025 (TRDOS 386 v2.0.10)
2584      <1>     ; 30/07/2022
2585      <1>     ; 29/07/2022 (TRDOS 386 Kernel v2.0.5)
2586      <1>     ; 16/10/2016

```

```

2587      <1>      ; 15/10/2016
2588      <1>      ; 23/03/2016
2589      <1>      ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)
2590      <1>      ; 01/08/2011 (DIR.ASM, 'proc_locate_current_dir_entry')
2591      <1>      ; 07/03/2010
2592      <1>      ; INPUT ->
2593      <1>      ; EAX = Directory Entry (Index) Number (< 65536)
2594      <1>      ; OUTPUT ->
2595      <1>      ; EDI = Directory Entry Address
2596      <1>      ; EAX = Cluster Number of Directory Buffer
2597      <1>      ; EBX = Directory Buffer Entry Offset
2598      <1>      ; ECX = DirBuff Valid Data identifier (CL)
2599      <1>      ; If CF = 0 and CL = 2 then
2600      <1>      ;     directory buffer modified and
2601      <1>      ;     must be written to disk.
2602      <1>      ; If CF = 0 and CL = 1 then
2603      <1>      ;     dir buffer has been written to disk, already.
2604      <1>      ; CF = 1 -> Error code in EAX (AL)
2605      <1>      ;
2606      <1>      ; (Modified registers: EAX, EDX, ECX, EBX, EDI)
2607      <1>
2608      <1>      loc_locate_current_dir_entry:
2609      <1>      push     esi
2610      <1>      mov      ecx, eax
2611      <1>      ;mov     edx, 32
2612      <1>      ; 29/07/2022
2613      <1>      sub      edx, edx
2614      <1>      mov      dl, 32
2615      <1>      mul      edx
2616      <1>      mov      [LCDE_ByteOffset], eax
2617      <1>      xor      ebx, ebx
2618      <1>      mov      bh, [Current_Drv]
2619      <1>      mov      al, [DirBuff_DRV]
2620      <1>      sub      al, 'A'
2621      <1>      mov      esi, Logical_DOSDisks
2622      <1>      add      esi, ebx
2623      <1>      cmp      bh, al
2624      <1>      ;jne     loc_lcde_reload_current_directory
2625      <1>      ; 29/07/2022
2626      <1>      je      short loc_lcde_cd1_check
2627      <1>      jmp     loc_lcde_reload_current_directory
2628      <1>      loc_lcde_cd1_check:
2629      <1>      ; 29/07/2022
2630      <1>      xor      eax, eax
2631      <1>      cmp      byte [Current_Dir_Level], 0
2632      <1>      ja      short loc_lcde_calc_dirbuff_cluster_offset
2633      <1>      ; 27/02/2016
2634      <1>      ; TRDOS v1 has bug here for FAT32 fs !
2635      <1>      ; (Root Directory Entries for FAT32 = 0)
2636      <1>      cmp      byte [esi+LD_FATType], 3 ; FAT32
2637      <1>      jnb     short loc_lcde_calc_dirbuff_cluster_offset
2638      <1>
2639      <1>      loc_lcde_cd1_check_FAT12_16:
2640      <1>      ; 29/07/2022
2641      <1>      ;xor     eax, eax
2642      <1>      mov      ax, [esi+LD_BPB+RootDirEnts]
2643      <1>      ;dec     ax
2644      <1>      dec      eax
2645      <1>      ;xor     dx, dx
2646      <1>      ;cmp     ax, cx ; cx = Directory Entry (Index) Number
2647      <1>      ; 29/07/2022
2648      <1>      cmp      eax, ecx
2649      <1>      jb      short loc_lcde_stc_12h_retn
2650      <1>      mov      [LCDE_EntryIndex], cx
2651      <1>      xor      eax, eax
2652      <1>      jmp     loc_lcde_check_dir_buffer_cluster
2653      <1>
2654      <1>      loc_lcde_stc_12h_retn:
2655      <1>      pop      esi
2656      <1>      mov      ebx, ecx
2657      <1>      mov      ecx, edx
2658      <1>      ; 16/10/2016 (12h -> 12)
2659      <1>      mov      eax, 12 ; No more files
2660      <1>      retn
2661      <1>
2662      <1>      loc_lcde_calc_dirbuff_cluster_offset:
2663      <1>      ;mov     bl, [esi+LD_BPB+SecPerClust]
2664      <1>      ;xor     bh, bh
2665      <1>      ;mov     ax, [esi+LD_BPB+BytesPerSec]
2666      <1>      ;mul     bx
2667      <1>      ;;or     dx, dx ; If bytes per cluster > 32KB it is invalid
2668      <1>      ;; 29/07/2022
2669      <1>      ;or     dl, dl
2670      <1>      ;jnz     short loc_lcde_invalid_format
2671      <1>      ; 29/07/2022
2672      <1>      xor      ebx, ebx
2673      <1>      mov      bl, [esi+LD_BPB+SecPerClust]
2674      <1>      ;mov     ax, [esi+LD_BPB+BytesPerSec]
2675      <1>      ; 19/12/2025
2676      <1>      mov      ax, 512
2677      <1>      mul      ebx
2678      <1>      mov      ecx, eax
2679      <1>      ;mov     cx, ax ; BYTES PER CLUSTER
2680      <1>      mov      eax, [LCDE_ByteOffset]
2681      <1>      ;sub     edx, edx
2682      <1>      div      ecx
2683      <1>      cmp      eax, 65535
2684      <1>      ja      short loc_lcde_invalid_format
2685      <1>
2686      <1>      ; cluster sequence number of directory (< 65536)
2687      <1>      mov      [LCDE_ClusterSN], ax
2688      <1>
2689      <1>      ;mov     ax, dx ; byte offset in cluster (directory buffer)
2690      <1>      ; 29/07/2022
2691      <1>      mov      eax, edx
2692      <1>      ;mov     bx, 32 ; 1 dir entry = 32 bytes
2693      <1>      mov      b1, 32
2694      <1>      ;sub     dx, dx ; 0
2695      <1>      ;div     bx
2696      <1>      sub      edx, edx
2697      <1>      div      ebx
2698      <1>      mov      [LCDE_EntryIndex], ax ; dir entry index/sequence number
2699      <1>      ; (in directory buffer/cluster)
2700      <1>      loc_lcde_get_current_sub_dir_fcluster:
2701      <1>      mov      eax, [Current_Dir_FCluster]
2702      <1>
2703      <1>      loc_lcde_get_next_cluster:
2704      <1>      cmp      word [LCDE_ClusterSN], 0
2705      <1>      jna      short loc_lcde_check_dir_buffer_cluster
2706      <1>      mov      [LCDE_Cluster], eax
2707      <1>      call     get_next_cluster
2708      <1>      jc      short loc_lcde_check_gnc_error
2709      <1>      dec      word [LCDE_ClusterSN]
2710      <1>      jmp     short loc_lcde_get_next_cluster

```

```

2711 <1>
2712 <1> loc_lcde_reload_current_directory:
2713 0000AFDD 51 <1> push ecx
2714 0000AFDE E8CEF6FFFF <1> call reload_current_directory
2715 0000AFE3 59 <1> pop ecx
2716 <1> ;jnc loc_lcde_cd1_check
2717 <1> ;pop esi
2718 <1> ;retn
2719 <1> ; 09/08/2022
2720 0000AFE4 727E <1> jc short loc_lcde_retn
2721 0000AFE6 E96CFFFFFF <1> jmp loc_lcde_cd1_check
2722 <1>
2723 <1> loc_lcde_invalid_format:
2724 <1> ; 15/10/2016 (0Bh -> 28)
2725 <1> ;mov eax, 28 ; Invalid Format !
2726 <1> ; 29/07/2022
2727 0000AFEB 29C0 <1> sub eax, eax
2728 0000AFED B01C <1> mov al, 28
2729 <1> loc_lcde_drive_not_ready_read_err:
2730 0000AFEF F9 <1> stc
2731 0000AFF0 5E <1> pop esi
2732 0000AFF1 C3 <1> retn
2733 <1>
2734 <1> loc_lcde_check_gnc_error:
2735 0000AFF2 09C0 <1> or eax, eax
2736 0000AFF4 75F9 <1> jnz short loc_lcde_drive_not_ready_read_err
2737 0000AFF6 66FF0D[3A810100] <1> dec word [LCDE_ClusterSN]
2738 0000AFFD 75EC <1> jnz short loc_lcde_invalid_format
2739 0000AFFF A1[3C810100] <1> mov eax, [LCDE_Cluster]
2740 <1>
2741 <1> loc_lcde_check_dir_buffer_cluster:
2742 0000B004 3B05[8E7E0100] <1> cmp eax, [DirBuff_Cluster]
2743 0000B00A 755A <1> jne short loc_lcde_load_dir_cluster
2744 0000B00C 803D[897E0100]00 <1> cmp byte [DirBuff_ValidData], 0
2745 0000B013 7726 <1> ja short loc_lcde_check_dir_buffer_cluster_next
2746 0000B015 803D[60770100]00 <1> cmp byte [Current_Dir_Level], 0
2747 0000B01C 775D <1> ja short loc_lcde_load_dir_cluster_0
2748 <1> ; 27/02/2016
2749 <1> ; TRDOS v1 has bug here for FAT32 fs !
2750 0000B01E 807E0303 <1> cmp byte [esi+LD_FATType], 3 ; FAT32
2751 0000B022 7357 <1> jnb short loc_lcde_load_dir_cluster_0
2752 <1> ;
2753 0000B024 0FB74E17 <1> movzx ecx, word [esi+LD_BPB+RootDirEnts]
2754 0000B028 6683C10F <1> add cx, 15 ; round up (16 entries per sector)
2755 <1> ;shr cx, 4 ; 1 sector contains 16 dir entries
2756 <1> ; 29/07/2022
2757 0000B02C C1E904 <1> shr ecx, 4
2758 0000B02F 8B4664 <1> mov eax, [esi+LD_ROOTBegin]
2759 0000B032 EB52 <1> jmp short loc_lcde_load_dir_cluster_1
2760 <1>
2761 <1> loc_lcde_validate_dirBuff:
2762 0000B034 C605[897E0100]01 <1> mov byte [DirBuff_ValidData], 1
2763 <1>
2764 <1> lcde_check_dir_buffer_cluster_next:
2765 0000B03B 0FB71D[38810100] <1> movzx ebx, word [LCDE_EntryIndex]
2766 0000B042 663B1D[8C7E0100] <1> cmp bx, [DirBuff_LastEntry]
2767 0000B049 77A0 <1> ja short loc_lcde_invalid_format
2768 <1> ;mov eax, 32
2769 <1> ; 29/07/2022
2770 0000B04B 31C0 <1> xor eax, eax
2771 0000B04D B020 <1> mov al, 32
2772 0000B04F F7E3 <1> mul ebx
2773 <1> ;or edx, edx
2774 <1> ;jnz short loc_lcde_invalid_format
2775 <1>
2776 0000B051 8F00000800 <1> mov edi, Directory_Buffer
2777 0000B056 01C7 <1> add edi, eax ; add entry offset to buffer address
2778 <1>
2779 <1> loc_lcde_dir_buffer_last_check:
2780 0000B058 A1[8E7E0100] <1> mov eax, [DirBuff_Cluster]
2781 0000B05D 0FB60D[897E0100] <1> movzx ecx, byte [DirBuff_ValidData]
2782 <1>
2783 <1> loc_lcde_retn:
2784 0000B064 5E <1> pop esi
2785 0000B065 C3 <1> retn
2786 <1>
2787 <1> loc_lcde_load_dir_cluster:
2788 <1> ;cmp byte [DirBuff_ValidData], 2
2789 <1> ;jne short loc_lcde_load_dir_cluster_n2
2790 0000B066 50 <1> push eax
2791 0000B067 E815FDFFFF <1> call save_directory_buffer
2792 0000B06C 58 <1> pop eax
2793 0000B06D 72F5 <1> jc short loc_lcde_retn
2794 <1>
2795 <1> loc_lcde_load_dir_cluster_n2:
2796 0000B06F C605[897E0100]00 <1> mov byte [DirBuff_ValidData], 0
2797 0000B076 A3[8E7E0100] <1> mov [DirBuff_Cluster], eax
2798 <1>
2799 <1> loc_lcde_load_dir_cluster_0:
2800 <1> ;sub eax, 2
2801 <1> ; 30/07/2022
2802 0000B07B 48 <1> dec eax
2803 0000B07C 48 <1> dec eax
2804 0000B07D 0FB64E13 <1> movzx ecx, byte [esi+LD_BPB+SecPerClust]
2805 0000B081 F7E1 <1> mul ecx
2806 0000B083 034668 <1> add eax, [esi+LD_DATABegin]
2807 <1>
2808 <1> loc_lcde_load_dir_cluster_1:
2809 0000B086 BB00000800 <1> mov ebx, Directory_Buffer
2810 <1> ; ecx = sector count
2811 0000B08B E853690000 <1> call disk_read
2812 0000B090 73A2 <1> jnc short loc_lcde_validate_dirBuff
2813 <1>
2814 <1> ; 15/10/2016
2815 <1> ; (Disk read error instead of drv not ready err)
2816 0000B092 B811000000 <1> mov eax, 17 ; Drive not ready or read error !
2817 0000B097 EBCB <1> jmp short loc_lcde_retn
2818 <1>
2819 <1> remove_file:
2820 <1> ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
2821 <1> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
2822 <1> ; 15/10/2016
2823 <1> ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
2824 <1> ; 10/04/2011 (FILE.ASM, 'proc_delete_file')
2825 <1> ; 09/08/2010
2826 <1> ; INPUT ->
2827 <1> ; EDI = Directory Buffer Entry Address
2828 <1> ; CX = Directory Buffer Entry Counter/Index
2829 <1> ; BL = Longname Entry Length
2830 <1> ; BH = Logical DOS Drive Number
2831 <1>
2832 0000B099 29C0 <1> sub eax, eax
2833 0000B09B 88FC <1> mov ah, bh
2834 0000B09D BE00010900 <1> mov esi, Logical_DOSDisks

```

```

2835 0000B0A2 01C6      <1>      add     esi, eax
2836                  <1>
2837                  <1> ; 19/12/2025
2838                  <1> %if 0
2839                  <1>      cmp     byte [esi+LD_FATType], 1
2840                  <1>      jnb     short loc_del_fat_file
2841                  <1>
2842                  <1>      cmp     byte [esi+LD_FSType], 0A1h
2843                  <1>      je      short loc_del_fs_file
2844                  <1>
2845                  <1> loc_del_file_invalid_format:
2846                  <1>      xor     ah, ah
2847                  <1>      ; 15/10/2016 (0Bh -> 28)
2848                  <1>      mov     al, 28 ; Invalid Format
2849                  <1>      stc
2850                  <1>      retn
2851                  <1>
2852                  <1> loc_del_fs_file:
2853                  <1>      ;call delete_fs_file
2854                  <1>      ;retn
2855                  <1>      ; 29/07/2022
2856                  <1>      jmp     delete_fs_file
2857                  <1> %endif
2858                  <1>
2859                  <1> loc_del_fat_file:
2860                  <1>      call    delete_directory_entry
2861                  <1>      jc      short loc_del_file_err_retn
2862                  <1>
2863                  <1> loc_delfile_unlink_cluster_chain:
2864                  <1>      call    truncate_cluster_chain
2865                  <1>      ;jc      short loc_del_file_err_retn
2866                  <1>
2867                  <1> loc_delfile_return:
2868                  <1> loc_del_file_err_retn:
2869                  <1>      retn
2870                  <1>
2871                  <1> delete_directory_entry:
2872                  <1>      ; 15/10/2016
2873                  <1>      ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
2874                  <1>      ; 01/08/2011 (DIR.ASM, 'proc_delete_directory_entry')
2875                  <1>      ; 10/04/2011
2876                  <1>      ; INPUT ->
2877                  <1>      ;      ESI = Logical Dos Drive Descripton Table Address
2878                  <1>      ;      EDI = Directory Buffer Entry Address
2879                  <1>      ;      CX = Directory Buffer Entry Counter/Index
2880                  <1>      ;      BL = Longname Entry Length
2881                  <1>      ; OUTPUT ->
2882                  <1>      ;      ESI = Logical dos drive descripton table address
2883                  <1>      ;      EAX = First cluster to be truncated/unlinked
2884                  <1>      ;      CF = 1 -> Error code in EAX (AL)
2885                  <1>      ;      CF = 0 & BH <> 0 -> LMDT write error (BH = 1)
2886                  <1>      ;      CF = 0 & BL <> 0 -> Long name delete error (BL = FFh)
2887                  <1>      ;
2888                  <1>      ; (EDI, EBX, ECX register contents will be changed)
2889                  <1>
2890                  <1>      mov     [DelFile_LNEL], bl
2891                  <1>      mov     [DelFile_EntryCounter], cx
2892                  <1>
2893                  <1>      mov     ax, [edi+20] ; First Cluster High word
2894                  <1>      shl     eax, 16
2895                  <1>      mov     ax, [edi+26] ; First Cluster Low word
2896                  <1>
2897                  <1>      mov     [DelFile_FCluster], eax
2898                  <1>
2899                  <1> loc_del_short_name:
2900                  <1>      mov     byte [edi], 0E5h ; Deleted sign
2901                  <1>
2902                  <1>      mov     byte [DirBuff_ValidData], 2
2903                  <1>      call    save_directory_buffer
2904                  <1>      jc      short loc_delete_direntry_err_return
2905                  <1>
2906                  <1> loc_del_long_name:
2907                  <1>      movzx   edx, byte [DelFile_LNEL]
2908                  <1>      or      dl, dl
2909                  <1>      jz      short loc_del_dir_entry_update_parent_dir_lm_date
2910                  <1>
2911                  <1>      mov     [DelFile_LNEL], dh ; 0
2912                  <1>
2913                  <1>      movzx   eax, word [DelFile_EntryCounter]
2914                  <1>      sub     eax, edx
2915                  <1>      jnc     short loc_del_long_name_continue
2916                  <1>      jc      short loc_del_dir_entry_update_parent_dir_lm_date
2917                  <1>
2918                  <1> ;loc_del_direntry_inv_data_return: ; 15/10/2016 (0Dh -> 29)
2919                  <1> ;      mov     eax, 29 ; 0Dh (TRDOS 8086) ; Invalid data
2920                  <1> ;      retn
2921                  <1>
2922                  <1> loc_del_long_name_continue:
2923                  <1>      ; AX = Directory Entry Number of the long name last entry
2924                  <1>      call    delete_longname
2925                  <1>      ;jc      short loc_delete_direntry_err_return
2926                  <1>
2927                  <1> loc_del_dir_entry_update_parent_dir_lm_date:
2928                  <1>      sbb     byte [DelFile_LNEL], 0 ; 0FFh if cf = 1
2929                  <1>
2930                  <1>      call    update_parent_dir_lmdt
2931                  <1>      mov     bh, 0
2932                  <1>      adc     bh, 0
2933                  <1>
2934                  <1>      mov     bl, byte [DelFile_LNEL]
2935                  <1>
2936                  <1> loc_delete_direntry_return:
2937                  <1>      mov     eax, [DelFile_FCluster]
2938                  <1> loc_delete_direntry_err_return:
2939                  <1>      retn
2940                  <1>
2941                  <1> rename_directory_entry:
2942                  <1>      ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
2943                  <1>      ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
2944                  <1>      ; 13/11/2017
2945                  <1>      ; 15/10/2016
2946                  <1>      ; 06/03/2016 (TRDOS 386 = TRDOS v2.0)
2947                  <1>      ; 01/08/2011 (DIR.ASM, 'proc_rename_directory_entry')
2948                  <1>      ; 19/11/2010
2949                  <1>      ; INPUT -> (Current Directory)
2950                  <1>      ;      CX = Directory Entry Number
2951                  <1>      ;      EAX = First Cluster number of file or directory
2952                  <1>      ;      EBX = Longname Length (dir entry count) (< 256)
2953                  <1>      ;      ESI = New file (or directory) name (no path).
2954                  <1>      ;      (ASCIIIZ string)
2955                  <1>      ; OUTPUT ->
2956                  <1>      ;      CF = 0 -> successfull
2957                  <1>      ;      CF = 1 -> error code in EAX (AL)
2958                  <1>

```

```

2959      ; (EAX, EBX, ECX, EDX, ESI, EDI will be changed)
2960
2961      ; 19/12/2025
2962      %if 0
2963      cmp     byte [Current_FATType], 0
2964      ja      short loc_rename_directory_entry
2965
2966      ; call rename_fs_file_or_directory
2967      ; retn
2968      ; 29/07/2022
2969      jmp     rename_fs_file_or_directory
2970
2971      %endif
2972
2973      loc_rename_directory_entry:
2974      mov     [DelFile_LNEL], bl
2975      mov     [DelFile_EntryCounter], cx
2976      mov     [DelFile_FCluster], eax
2977
2978      movzx   eax, cx
2979      call    locate_current_dir_entry
2980      jnc     short loc_rename_direntry_check_fcluster
2981
2982      loc_rename_direntry_pop_retn:
2983      retn
2984
2985      loc_rename_direntry_pop_invd_retn:
2986      ; 29/07/2022
2987      ; stc
2988      loc_rename_direntry_invd_retn:
2989      ; 15/10/2016 (0Dh -> 29)
2990      ; mov     eax, 29 ; Invalid data
2991      ; 29/07/2022
2992      sub     eax, eax
2993      mov     al, 29
2994      stc
2995      loc_rename_retn:
2996      retn
2997
2998      loc_rename_direntry_check_fcluster:
2999      mov     dx, [edi+20] ; First cluster HW
3000      shl     edx, 16 ; 13/11/2017
3001      mov     dx, [edi+26] ; First cluster LW
3002      cmp     edx, [DelFile_FCluster]
3003      jne     short loc_rename_direntry_pop_invd_retn
3004      ; ESI = New file (or directory) name. (ASCIIz string)
3005      ; 06/03/2016
3006      ; TRDOS v2 - NOTE: 'convert_file_name' procedure
3007      ; has been modified for eliminating following situation.
3008      ;
3009      ; TRDOS v1 - NOTE: If file/dir name is more than 11 bytes
3010      ; without a dot, attributes (edi+11) byte will be overwritten !
3011      ; (Dot file name input must be proper for 11 byte dir entry
3012      ; type file name output.)
3013      call    convert_file_name
3014
3015      mov     byte [DirBuff_ValidData], 2
3016      call    save_directory_buffer
3017      jc      short loc_rename_retn
3018
3019      loc_rename_direntry_del_ln:
3020      movzx   edx, byte [DelFile_LNEL]
3021      or      dl, dl
3022      jz      short loc_rename_direntry_update_parent_dir_lm_date
3023
3024      movzx   eax, word [DelFile_EntryCounter]
3025      sub     eax, edx
3026      jc      short loc_rename_direntry_invd_retn
3027
3028      loc_rename_direntry_del_ln_continue:
3029      ; EAX = Directory Entry Number of the long name last entry
3030      call    delete_longname
3031
3032      loc_rename_direntry_update_parent_dir_lm_date:
3033      call    update_parent_dir_lmdt
3034      xor     eax, eax
3035      retn
3036
3037      move_source_file_to_destination_file:
3038      ; 19/12/2025
3039      ; 11/08/2025 (TRDOS 386 kernel v2.0.10)
3040      ; 07/08/2022
3041      ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
3042      ; 15/10/2016
3043      ; 11/03/2016
3044      ; 10/03/2016 (TRDOS 386 = TRDOS v2.0)
3045      ; 01/08/2011 (FILE.ASM)
3046      ; 04/08/2010
3047      ;
3048      ; Phase 1 -> Check destination file,
3049      ; 'not found' is required
3050      ; Phase 2 -> Check source file
3051      ; 'found' and proper attributes is required
3052      ; Phase 3 -> Make destination directory entry,
3053      ; add new dir cluster or section if it is required
3054      ; Phase 4 -> Delete source directory entry.
3055      ; cf = 1 causes to return before the phase 4.
3056      ; (source file protection against any possible errors)
3057      ;
3058      ; 08/05/2011 major modification
3059      ; -> destination file deleting is removed
3060      ; for msdos move/rename compatibility.
3061      ; (Access denied error will return if
3062      ; the destination file is found...)
3063      ; INPUT ->
3064      ; ESI = Source File Pathname (AsciiZ)
3065      ; EDI = Destination File Pathname (AsciiZ)
3066      ; AL = 0 --> Interrupt (System call)
3067      ; AL > 0 --> Command Interpreter (Question)
3068      ; AL = 1 --> Question Phase
3069      ; AL = 2 --> Progress Phase
3070      ; OUTPUT ->
3071      ; cf = 0 -> OK
3072      ; EAX = Destination directory first cluster
3073      ; ESI = Logical DOS drive description table
3074      ; EBX = Destination file structure offset
3075      ; CX = 0 (CX > 0 --> calculate free space error)
3076      ; cf = 1 -> Error code in EAX (AL)
3077      ;
3078      ; (EDX, ECX, EBX, ESI, EDI will be changed)
3079
3080      cmp     al, 2
3081      jne     msftdf_df2_check_directory
3082      ; 29/07/2022
3083      jne     short msftdf

```



```

3083 0000B18D E96D010000    <1>      jmp      msftdf_df2_check_directory
3084                                <1> msftdf:
3085 0000B192 A2[4E820100]    <1>      mov     [move_cmd_phase], al
3086                                <1>
3087                                <1> msftdf_parse_sf_path:
3088                                <1>      ; ESI = ASCIIZ pathname (Source)
3089 0000B197 57              <1>      push    edi
3090 0000B198 BF[4C810100]    <1>      mov     edi, SourceFile_Drv
3091 0000B19D E8CCF7FFFF    <1>      call    parse_path_name
3092 0000B1A2 5E              <1>      pop     esi
3093 0000B1A3 7211          <1>      jc      short msftdf_psf_retn
3094                                <1>
3095                                <1> msftdf_parse_df_path:
3096                                <1>      ; ESI = ASCIIZ pathname (Destination)
3097 0000B1A5 BF[CC810100]    <1>      mov     edi, DestinationFile_Drv
3098 0000B1AA E8BFF7FFFF    <1>      call    parse_path_name
3099 0000B1AF 7306          <1>      jnc     short msftdf_check_sf_drv
3100                                <1>
3101 0000B1B1 3C01          <1>      cmp     al, 1 ; File or directory name is not existing
3102 0000B1B3 7602          <1>      jna     short msftdf_check_sf_drv
3103                                <1>
3104                                <1> msftdf_stc_retn:
3105 0000B1B5 F9              <1>      stc
3106                                <1> msftdf_psf_retn:
3107 0000B1B6 C3              <1>      retn
3108                                <1>
3109                                <1> msftdf_check_sf_drv:
3110 0000B1B7 A0[4C810100]    <1>      mov     al, [SourceFile_Drv]
3111                                <1>
3112                                <1> msftdf_check_df_drv:
3113 0000B1BC 8A15[CC810100]    <1>      mov     dl, [DestinationFile_Drv]
3114                                <1>
3115                                <1> msftdf_compare_sf_df_drv:
3116                                <1>      ; 11/08/2025
3117                                <1>      ;sub     ebx, ebx
3118                                <1>      ;mov     bh, [Current_Drv]
3119 0000B1C2 38C2          <1>      cmp     dl, al
3120 0000B1C4 7408          <1>      je      short msftdf_check_sf_df_drv_ok
3121                                <1>
3122                                <1> msftdf_not_same_drv:
3123                                <1>      ; DL = source file's drive number
3124 0000B1C6 88C6          <1>      mov     dh, al ; destination file's drive number
3125                                <1>      ; 15/10/2016 (11h -> 21)
3126                                <1>      ;mov     eax, 21 ; Not the same drive
3127                                <1>      ; 29/07/2022
3128 0000B1C8 29C0          <1>      sub     eax, eax
3129 0000B1CA B015          <1>      mov     al, 21
3130 0000B1CC F9              <1>      stc
3131 0000B1CD C3              <1>      retn
3132                                <1>
3133                                <1> msftdf_check_sf_df_drv_ok:
3134 0000B1CE 8815[4F820100]    <1>      mov     [msftdf_sf_df_drv], dl
3135                                <1>
3136                                <1>      sub     eax, eax
3137 0000B1D6 88D4          <1>      mov     ah, dl
3138 0000B1D8 0500010900    <1>      add     eax, Logical_DOSDisks
3139 0000B1DD A3[50820100]    <1>      mov     [msftdf_drv_offset], eax
3140                                <1>
3141                                <1>      ;cmp     dl, bh ; byte [Current_Drv]
3142                                <1>      ; 11/08/2025
3143 0000B1E2 3A15[62770100]    <1>      cmp     dl, [Current_Drv]
3144 0000B1E8 7407          <1>      je      short msftdf_df_check_directory
3145                                <1>
3146                                <1> msftdf_change_drv:
3147 0000B1EA E8F1C4FFFF    <1>      call    change_current_drive
3148 0000B1EF 726C          <1>      jc      short msftdf_df_error_retn
3149                                <1>
3150                                <1> msftdf_check_destination_file:
3151                                <1> msftdf_df_check_directory:
3152 0000B1F1 BE[CD810100]    <1>      mov     esi, DestinationFile_Directory
3153 0000B1F6 803E20          <1>      cmp     byte [esi], 20h
3154 0000B1F9 760F          <1>      jna     short msftdf_df_find_1
3155                                <1>
3156                                <1> msftdf_df_change_directory:
3157 0000B1FB FE05[7E2E0100]    <1>      inc     byte [Restore_CDIR]
3158 0000B201 30E4          <1>      xor     ah, ah ; CD_COMMAND sign -> 0
3159 0000B203 E8D3F1FFFF    <1>      call    change_current_directory
3160 0000B208 7253          <1>      jc      short msftdf_df_error_retn
3161                                <1>
3162                                <1> ;msftdf_df_change_prompt_dir_string:
3163                                <1> ; call     change_prompt_dir_string
3164                                <1>
3165                                <1> msftdf_df_find_1:
3166 0000B20A BE[0E820100]    <1>      mov     esi, DestinationFile_Name
3167 0000B20F 803E20          <1>      cmp     byte [esi], 20h
3168 0000B212 7631          <1>      jna     short msftdf_df_copy_sf_name
3169                                <1>
3170                                <1> msftdf_df_find_2:
3171                                <1> ;xor     ax, ax ; DestinationFile_AttributesMask -> any/zero
3172                                <1> ; 11/08/2025
3173 0000B214 31C0          <1>      xor     eax, eax
3174 0000B216 E84FD7FFFF    <1>      call    find_first_file
3175                                <1> ;jnc     msftdf_permission_denied_retn
3176                                <1> ; 29/07/2022
3177 0000B21B 7205          <1>      jc      short msftdf_df_check_error_code
3178 0000B21D E98B000000    <1>      jmp     msftdf_permission_denied_retn
3179                                <1>
3180                                <1> msftdf_df_check_error_code:
3181                                <1> ;cmp     eax, 2 ; File not found error
3182 0000B222 3C02          <1>      cmp     al, 2
3183 0000B224 7536          <1>      jne     short msftdf_df_stc_retn
3184                                <1>
3185                                <1> msftdf_df_check_fname:
3186                                <1> ; 15/10/2016
3187 0000B226 BE[0E820100]    <1>      mov     esi, DestinationFile_Name ; *
3188 0000B22B E899DAFFFF    <1>      call    check_filename
3189 0000B230 7307          <1>      jnc     short msftdf_convert_df_direntry_name
3190                                <1> ; invalid file name chars !
3191 0000B232 B81A000000    <1>      mov     eax, ERR_INV_FILE_NAME ; 26
3192 0000B237 EB23          <1>      jmp     short msftdf_df_stc_retn
3193                                <1>
3194                                <1> msftdf_convert_df_direntry_name:
3195                                <1> ;mov     esi, DestinationFile_Name ; *
3196 0000B239 BF[1E820100]    <1>      mov     edi, DestinationFile_DirEntry
3197 0000B23E E861F6FFFF    <1>      call    convert_file_name
3198 0000B243 EB19          <1>      jmp     short msftdf_restore_current_dir_1
3199                                <1>
3200                                <1> msftdf_df_copy_sf_name:
3201 0000B245 89F7          <1>      mov     edi, esi
3202 0000B247 57              <1>      push    edi
3203 0000B248 BE[8E810100]    <1>      mov     esi, SourceFile_Name
3204                                <1> ;mov     ecx, 12
3205                                <1> ; 29/07/2022
3206 0000B24D 29C9          <1>      sub     ecx, ecx

```

```

3207 0000B24F B10C      <1>      mov     cl, 12
3208                    <1> msftdf_df_copy_sf_name_loop:
3209 0000B251 AC          <1>      lodsb
3210 0000B252 AA          <1>      stosb
3211 0000B253 08C0       <1>      or      al, al
3212 0000B255 7402       <1>      jz      short msftdf_df_copy_sf_name_ok
3213 0000B257 E2F8       <1>      loop    msftdf_df_copy_sf_name_loop
3214                    <1> msftdf_df_copy_sf_name_ok:
3215 0000B259 5E          <1>      pop     esi
3216 0000B25A EBB8       <1>      jmp     short msftdf_df_find_2
3217                    <1>
3218                    <1> msftdf_df_stc_retn:
3219 0000B25C F9          <1>      stc
3220                    <1> msftdf_restore_cdir_failed:
3221                    <1> msftdf_df_error_retn:
3222 0000B25D C3          <1>      retn
3223                    <1>
3224                    <1> msftdf_restore_current_dir_1:
3225 0000B25E 803D[7E2E0100]00 <1>      cmp     byte [Restore_CDIR], 0
3226 0000B265 760D       <1>      jna     short msftdf_sf_check_directory
3227 0000B267 8B35[50820100] <1>      mov     esi, [msftdf_drv_offset]
3228 0000B26D E81FC5FFFF <1>      call    restore_current_directory
3229 0000B272 72E9       <1>      jc      short msftdf_restore_cdir_failed
3230                    <1>
3231                    <1> msftdf_sf_check_directory:
3232 0000B274 BE[4D810100] <1>      mov     esi, SourceFile_Directory
3233 0000B279 803E20     <1>      cmp     byte [esi], 20h
3234 0000B27C 760F       <1>      jna     short msftdf_sf_find
3235                    <1> msftdf_sf_change_directory:
3236 0000B27E FE05[7E2E0100] <1>      inc     byte [Restore_CDIR]
3237 0000B284 30E4       <1>      xor     ah, ah ; CD_COMMAND sign -> 0
3238 0000B286 E850F1FFFF <1>      call    change_current_directory
3239 0000B28B 7225       <1>      jc      short msftdf_return
3240                    <1>
3241                    <1> ;msftdf_sf_change_prompt_dir_string:
3242                    <1> ;      call    change_prompt_dir_string
3243                    <1>
3244                    <1> msftdf_sf_find:
3245 0000B28D BE[8E810100] <1>      mov     esi, SourceFile_Name ; Offset 66
3246 0000B292 66B80018   <1>      mov     ax, 1800h ; Only files
3247 0000B296 E8CFD6FFFF <1>      call    find_first_file
3248 0000B29B 7215       <1>      jc      short msftdf_return
3249                    <1>
3250                    <1> msftdf_sf_ambgfn_check:
3251 0000B29D 6609D2     <1>      or      dx, dx ; Ambiguous filename chars used sign (dx>0)
3252 0000B2A0 7406       <1>      jz      short msftdf_sf_found
3253                    <1>
3254                    <1> msftdf_ambiguous_file_name_error:
3255                    <1> ;mov     eax, 2 ; File not found error
3256                    <1> ; 29/07/2022
3257 0000B2A2 29C0       <1>      sub     eax, eax
3258 0000B2A4 B002       <1>      mov     al, 2
3259 0000B2A6 F9          <1>      stc
3260 0000B2A7 C3          <1>      retn
3261                    <1>
3262                    <1> msftdf_sf_found:
3263 0000B2A8 80E31F     <1>      and     bl, 1Fh ; Attributes, D-V-S-H-R
3264 0000B2AB 7415       <1>      jz      short msftdf_save_sf_structure
3265                    <1>
3266                    <1> msftdf_permission_denied_retn:
3267                    <1> ;mov     eax, 05h ; Access (Permission) denied !
3268                    <1> ; 29/07/2022
3269 0000B2AD 29C0       <1>      sub     eax, eax
3270 0000B2AF B005       <1>      mov     al, 5
3271 0000B2B1 F9          <1>      stc
3272                    <1> msftdf_rest_cdir_err_retn:
3273                    <1> msftdf_return:
3274 0000B2B2 C3          <1>      retn
3275                    <1>
3276                    <1> msftdf_phase_1_return:
3277 0000B2B3 31C0       <1>      xor     eax, eax
3278 0000B2B5 A2[4E820100] <1>      mov     [move_cmd_phase], al ; 0
3279 0000B2BA FEC0       <1>      inc     al ; mov al, 1
3280 0000B2BC BB[FEB20000] <1>      mov     ebx, msftdf_df2_check_directory
3281                    <1> ;mov     edx, 0FFFFFFFh
3282 0000B2C1 C3          <1>      retn
3283                    <1>
3284                    <1> msftdf_save_sf_structure:
3285 0000B2C2 BE[54800100] <1>      mov     esi, FindFile_DirEntry
3286 0000B2C7 BF[9E810100] <1>      mov     edi, SourceFile_DirEntry
3287                    <1> ;mov     ecx, 8
3288                    <1> ; 29/07/2022
3289 0000B2CC 29C9       <1>      sub     ecx, ecx
3290 0000B2CE B108       <1>      mov     cl, 8
3291 0000B2D0 F3A5       <1>      rep     movsd
3292                    <1>
3293                    <1> msftdf_df_copy_sf_parameters:
3294                    <1> ;mov     esi, 11
3295                    <1> ;mov     edi, esi
3296                    <1> ;add     esi, SourceFile_DirEntry
3297                    <1> ;add     edi, DestinationFile_DirEntry
3298                    <1> ; 11/08/2025
3299 0000B2D2 BE[A9810100] <1>      mov     esi, SourceFile_DirEntry+11
3300 0000B2D7 BF[29820100] <1>      mov     edi, DestinationFile_DirEntry+11
3301                    <1> ;mov     ecx, 21
3302 0000B2DC B115       <1>      mov     cl, 21
3303 0000B2DE F3A4       <1>      rep     movsb
3304                    <1>
3305                    <1> msftdf_restore_current_dir_2:
3306 0000B2E0 803D[7E2E0100]00 <1>      cmp     byte [Restore_CDIR], 0
3307 0000B2E7 760D       <1>      jna     short msftdf_df2_check_move_cmd_phase
3308 0000B2E9 8B35[50820100] <1>      mov     esi, [msftdf_drv_offset]
3309 0000B2EF E89DC4FFFF <1>      call    restore_current_directory
3310 0000B2F4 72BC       <1>      jc      short msftdf_rest_cdir_err_retn
3311                    <1>
3312                    <1> msftdf_df2_check_move_cmd_phase:
3313 0000B2F6 803D[4E820100]01 <1>      cmp     byte [move_cmd_phase], 1
3314 0000B2FD 74B4       <1>      je      short msftdf_phase_1_return
3315                    <1>
3316                    <1> msftdf_df2_check_directory:
3317 0000B2FF BE[CD810100] <1>      mov     esi, DestinationFile_Directory
3318 0000B304 803E20     <1>      cmp     byte [esi], 20h
3319 0000B307 760F       <1>      jna     short msftdf_make_dfde_locate_fffe_on_directory
3320                    <1> msftdf_df2_change_directory:
3321 0000B309 FE05[7E2E0100] <1>      inc     byte [Restore_CDIR]
3322 0000B30F 30E4       <1>      xor     ah, ah ; CD_COMMAND sign -> 0
3323 0000B311 E8C5F0FFFF <1>      call    change_current_directory
3324 0000B316 729A       <1>      jc      short msftdf_return
3325                    <1>
3326                    <1> ;msftdf_df2_change_prompt_dir_string:
3327                    <1> ;      call    change_prompt_dir_string
3328                    <1>
3329                    <1> msftdf_make_dfde_locate_fffe_on_directory:
3330                    <1> ; Current directory fcluster <> Directory buffer cluster

```

```

3331      <1>      ; Current directory will be reloaded by
3332      <1>      ; 'locate_current_dir_file' procedure
3333      <1>      ;
3334      <1>      ;xor     ax, ax
3335      0000B318 31C0      <1>      xor     eax, eax
3336      0000B31A 89C1      <1>      mov     ecx, eax
3337      0000B31C 6649      <1>      dec     cx ; FFFFh
3338      <1>      ; CX = FFFFh -> find first deleted or free entry
3339      <1>      ; ESI would be ASCIIZ filename address if the call
3340      <1>      ; would not be for first free or deleted dir entry
3341      0000B31E E8D5F2FFFF <1>      call    locate_current_dir_file
3342      <1>      ; 07/08/2022
3343      0000B323 7328      <1>      jnc     short msftdf_make_dfde_set_ff_dir_entry
3344      <1>
3345      <1>      ;cmp     eax, 2
3346      0000B325 3C02      <1>      cmp     al, 2
3347      0000B327 7520      <1>      jne     short msftdf_error_retn
3348      <1>
3349      <1>      msftdf_add_new_dir_entry_check_fs:
3350      0000B329 8B35[50820100] <1>      mov     esi, [msftdf_drv_offset]
3351      0000B32F A1[8E7E0100] <1>      mov     eax, [DirBuff_Cluster]
3352      <1>
3353      <1>      ; 19/12/2025
3354      <1>      %if 0
3355      <1>      cmp     byte [esi+LD_FATType], 0
3356      <1>      ja      short msftdf_add_new_subdir_cluster
3357      <1>
3358      <1>      msftdf_add_new_fs_subdir_section:
3359      <1>      ;CL=0, CH=E5h --> deleted entry, CH=0 --> free entry
3360      <1>      ;xor     cx, cx
3361      <1>      xor     ch, ch ; cx = 0 --> add a new subdir section
3362      <1>      call    add_new_fs_section
3363      <1>      jc      short msftdf_dsfd_error_retn
3364      <1>      ;mov     [createfile_LastDirCluster], eax
3365      <1>
3366      <1>      call    load_FS_sub_directory
3367      <1>      ;mov     ebx, Directory_Buffer
3368      <1>      jnc     short msftdf_add_new_fs_subdir_section_ok
3369      <1>      retn
3370      <1>      %endif
3371      <1>
3372      <1>      msftdf_add_new_subdir_cluster:
3373      0000B334 E8C3140000 <1>      call    add_new_cluster
3374      0000B339 720F      <1>      jc      short msftdf_dsfd_error_retn
3375      <1>
3376      <1>      ;mov     [createfile_LastDirCluster], eax
3377      <1>
3378      <1>      call    load_FAT_sub_directory
3379      0000B340 7309      <1>      jnc     short msftdf_add_new_subdir_cluster_ok
3380      <1>      ; EBX = Directory buffer address
3381      <1>
3382      <1>      msftdf_ansdc_update_parent_dir_lmdt:
3383      <1>      msftdf_make_dfde_err_upd_pdir_lmdt:
3384      0000B342 50      <1>      push    eax
3385      0000B343 E8C2FAFFFF <1>      call    update_parent_dir_lmdt
3386      0000B348 58      <1>      pop     eax
3387      <1>
3388      <1>      msftdf_error_retn:
3389      0000B349 F9      <1>      stc
3390      <1>      msftdf_dsfd_restore_cdir_failed:
3391      <1>      msftdf_dsfd_error_retn:
3392      0000B34A C3      <1>      retn
3393      <1>
3394      <1>      msftdf_add_new_fs_subdir_section_ok:
3395      <1>      msftdf_add_new_subdir_cluster_ok:
3396      0000B34B 89DF      <1>      mov     edi, ebx ; Directory buffer address
3397      <1>
3398      <1>      msftdf_make_dfde_set_ff_dir_entry:
3399      0000B34D 8B15[5C770100] <1>      mov     edx, [Current_Dir_FCluster]
3400      0000B353 8915[B4820100] <1>      mov     [createfile_FFCluster], edx
3401      <1>      ; EDI = Directory entry offset
3402      0000B359 BE[1E820100] <1>      mov     esi, DestinationFile_DirEntry
3403      <1>      ;mov     ecx, 8
3404      <1>      ; 29/07/2022
3405      0000B35E 29C9      <1>      sub     ecx, ecx
3406      0000B360 B108      <1>      mov     cl, 8
3407      0000B362 F3A5      <1>      rep     movsd
3408      <1>
3409      0000B364 C605[897E0100]02 <1>      mov     byte [DirBuff_ValidData], 2
3410      0000B36B E811FAFFFF <1>      call    save_directory_buffer
3411      0000B370 72D0      <1>      jc      short msftdf_make_dfde_err_upd_pdir_lmdt
3412      <1>
3413      <1>      msftdf_make_dfde_update_pdir_lmdt:
3414      0000B372 E893FAFFFF <1>      call    update_parent_dir_lmdt
3415      <1>
3416      <1>      msftdf_dsfd_restore_current_dir_1:
3417      0000B377 803D[7E2E0100]00 <1>      cmp     byte [Restore_CDIR], 0
3418      0000B37E 760D      <1>      jna     short msftdf_dsfd_check_directory
3419      0000B380 8B35[50820100] <1>      mov     esi, [msftdf_drv_offset]
3420      0000B386 E806C4FFFF <1>      call    restore_current_directory
3421      0000B38B 72BD      <1>      jc      short msftdf_dsfd_restore_cdir_failed
3422      <1>
3423      <1>      msftdf_dsfd_check_directory:
3424      0000B38D BE[4D810100] <1>      mov     esi, SourceFile_Directory
3425      0000B392 803E20      <1>      cmp     byte [esi], 20h
3426      0000B395 760F      <1>      jna     short msftdf_dsfd_find_file
3427      <1>
3428      <1>      msftdf_dsfd_change_directory:
3429      0000B397 FE05[7E2E0100] <1>      inc     byte [Restore_CDIR]
3430      0000B39D 28E4      <1>      sub     ah, ah ; CD_COMMAND sign -> 0
3431      0000B39F E837F0FFFF <1>      call    change_current_directory
3432      0000B3A4 72A4      <1>      jc      short msftdf_dsfd_error_retn
3433      <1>
3434      <1>      ;msftdf_dsfd_sf_change_prompt_dir_string:
3435      <1>      ; call    change_prompt_dir_string
3436      <1>
3437      <1>      msftdf_dsfd_find_file:
3438      0000B3A6 BE[8E810100] <1>      mov     esi, SourceFile_Name ; Offset 66
3439      0000B3AB 668B460E <1>      mov     ax, [esi+14] ; 80 -> SourceFile_AttributesMask
3440      0000B3AF E8B6D5FFFF <1>      call    find_first_file
3441      0000B3B4 7294      <1>      jc      short msftdf_dsfd_error_retn
3442      <1>
3443      <1>      msftdf_dsfd_delete_direntry:
3444      0000B3B6 8B35[50820100] <1>      mov     esi, [msftdf_drv_offset]
3445      <1>
3446      <1>      ; 19/12/2025
3447      <1>      %if 0
3448      <1>      cmp     byte [esi+LD_FATType], 0
3449      <1>      ja      short msftdf_delete_FAT_direntry
3450      <1>
3451      <1>      xor     bl, bl
3452      <1>      ; BL = 0 -> File
3453      <1>      ; EDI -> Directory buffer entry offset/address
3454      <1>      call    delete_fs_directory_entry

```

```

3455         jnc     short msftdf_dsfde_restore_current_dir_2
3456         retn
3457     %endif
3458
3459     <1> msftdf_delete_FAT_direntry:
3460     0000B3BC 8A1D[51800100]    <1>     mov     bl, [FindFile_LongNameEntryLength]
3461     0000B3C2 668B0D[7C800100]    <1>     mov     cx, [FindFile_DirEntryNumber]
3462     <1>     ; ESI = Logical DOS drive description table address
3463     <1>     ; EDI = Directory buffer entry offset/address
3464     0000B3C9 E8E3FCFFFF    <1>     call    delete_directory_entry
3465     0000B3CE 721C    <1>     jc     short msftdf_retn
3466     <1>
3467     <1> msftdf_dsfde_restore_current_dir_2:
3468     0000B3D0 803D[7E2E0100]00    <1>     cmp     byte [Restore_CDIRE], 0
3469     0000B3D7 7607    <1>     jna     short msftdf_new_dir_fcluster_retn
3470     <1>     ;mov     esi, [msftdf_drv_offset]
3471     0000B3D9 E8B3C3FFFF    <1>     call    restore_current_directory
3472     0000B3DE 720C    <1>     jc     short msftdf_retn
3473     <1>
3474     <1> msftdf_new_dir_fcluster_retn:
3475     0000B3E0 31C9    <1>     xor     ecx, ecx
3476     0000B3E2 A1[B4820100]    <1>     mov     eax, [createfile_FFcluster]
3477     0000B3E7 BB[CC810100]    <1>     mov     ebx, DestinationFile_Drv
3478     <1>
3479     <1> msftdf_retn:
3480     0000B3EC C3    <1>     retn
3481     <1>
3482     <1> copy_source_file_to_destination_file:
3483     <1>     ; 19/12/2025 (TRDOS 386 kernel v2.0.10)
3484     <1>     ; 31/08/2024
3485     <1>     ; 30/08/2024
3486     <1>     ; 29/08/2024
3487     <1>     ; 26/08/2024
3488     <1>     ; 25/08/2024 (TRDOS 386 v2.0.9)
3489     <1>     ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
3490     <1>     ; 17/10/2016
3491     <1>     ; 16/10/2016
3492     <1>     ; 15/10/2016
3493     <1>     ; 30/03/2016, 31/03/2016
3494     <1>     ; 24/03/2016, 25/03/2016, 28/03/2016
3495     <1>     ; 21/03/2016, 22/03/2016, 23/03/2016
3496     <1>     ; 16/03/2016, 17/03/2016, 18/03/2016
3497     <1>     ; 15/03/2016 (TRDOS 386 = TRDOS v2.0)
3498     <1>     ; 02/09/2011 (FILE.ASM 'copy_source_file_to_destination_file')
3499     <1>     ; 01/08/2010 - 18/05/2011
3500     <1>
3501     <1>     ; Command Interpreter phase 1 enter ->
3502     <1>     ;     AL = 1 -> Caller is command interpreter
3503     <1>     ;     AL = 2 -> The second call, re-enter/continue
3504     <1>     ; Phase 1 -> Check source file
3505     <1>     ;     'found' is required
3506     <1>     ; Phase 2 -> Check destination file,
3507     <1>     ;     save 'found' or 'not found' status
3508     <1>     ;     'permission denied' error will be return
3509     <1>     ;     if attributes have not for ordinary file
3510     <1>     ;     without readonly attribute
3511     <1>     ; Command Interpreter phase 1 return ->
3512     <1>     ;     DH = Source file attributes
3513     <1>     ;     DL = Destination file found status
3514     <1>     ;     EAX = 0
3515     <1>     ; Command Interpreter phase 2 enter ->
3516     <1>     ;     AL = 2 -> Continue from the last position
3517     <1>     ;     AH =
3518     <1>     ; Phase 3 -> Load source file or use read/write cluster method
3519     <1>     ; Phase 4 -> Create destination file if it is not found
3520     <1>     ; Phase 5 -> Open destination file
3521     <1>     ; Phase 6 -> Read from source and write to destination
3522     <1>     ; Phase 7 -> Unload source file, if it is loaded at memory
3523     <1>     ;     cf = 1 causes to return before the phase 7
3524     <1>     ;     but loaded file will be unloaded
3525     <1>     ;     (allocated memory block will be deallocated)
3526     <1>
3527     <1>     ; INPUT ->
3528     <1>     ;     ESI = Source File Pathname (Asciiz)
3529     <1>     ;     EDI = Destination File Pathname (Asciiz)
3530     <1>     ;     AL = 0 --> Interrupt (System call)
3531     <1>     ;     AL > 0 --> Command Interpreter (Question)
3532     <1>     ;     AL = 1 --> Question Phase
3533     <1>     ;     AL = 2 --> Progress Phase
3534     <1>
3535     <1>     ; OUTPUT ->
3536     <1>     ;     cf = 0 -> OK ; (*)
3537     <1>     ;     EAX = Destination file first cluster
3538     <1>
3539     <1>     ; 31/08/2024 - TRDOS v2.0.9 ; (*)
3540     <1>     ; obsolete ;CL > 0 if there is file reading error before EOF
3541     <1>     ;     ; (incomplete copy)
3542     <1>     ;     ;CH > 0 if file is (full) loaded at memory
3543     <1>
3544     <1>     ;     cf = 1 -> Error code in AL (EAX) ; (*)
3545     <1>
3546     <1>     ; (EBX, ECX, ESI, EDI register contents will be changed)
3547     <1>
3548     0000B3ED 3C02    <1>     cmp     al, 2
3549     <1>     ;je     csftdf2_check_cdrv
3550     <1>     ; 29/07/2022
3551     0000B3EF 7205    <1>     jnb     short csftdf_ph1
3552     <1>     ;jmp     csftdf2_check_cdrv
3553     0000B3F1 E943020000    <1>     jmp     csftdf_ph2
3554     <1>
3555     <1> ; Phase 1
3556     <1> ; 29/07/2022
3557     <1> csftdf_ph1:
3558     0000B3F6 A2[74820100]    <1>     mov     byte [copy_cmd_phase], al
3559     <1>
3560     0000B3FB 57    <1>     push    edi ; *
3561     <1>
3562     <1> csftdf_parse_sf_path:
3563     0000B3FC BF[4C810100]    <1>     mov     edi, SourceFile_Drv
3564     0000B401 E868F5FFFF    <1>     call    parse_path_name
3565     0000B406 721C    <1>     jc     short csftdf_parse_sf_path_failed
3566     <1>
3567     <1> csftdf_parse_df_path:
3568     0000B408 5E    <1>     pop     esi ; * (pushed edi)
3569     <1>
3570     <1> csftdf_sf_check_filename_exists:
3571     0000B409 803D[8E810100]21    <1>     cmp     byte [SourceFile_Name], 21h
3572     0000B410 7214    <1>     jnb     short csftdf_sf_file_not_found_error
3573     <1>
3574     0000B412 BF[CC810100]    <1>     mov     edi, DestinationFile_Drv
3575     0000B417 E852F5FFFF    <1>     call    parse_path_name
3576     0000B41C 730E    <1>     jnc     short csftdf_check_sf_cdrv
3577     <1>
3578     0000B41E 3C01    <1>     cmp     al, 1 ; File or directory name is not existing

```

```

3579 0000B420 760A      <1>      jna      short csftdf_check_sf_cdrv
3580                  <1>
3581                  <1> csftdf_parse_df_path_failed:
3582 0000B422 F9         <1>      stc
3583                  <1> csftdf_sf_error_retn:
3584 0000B423 C3         <1>      retn
3585                  <1>
3586                  <1> csftdf_parse_sf_path_failed:
3587 0000B424 5F         <1>      pop      edi ; *
3588                  <1>      ; jmp      short csftdf_sf_error_retn
3589                  <1>      ; 25/08/2024
3590 0000B425 C3         <1>      retn
3591                  <1>
3592                  <1> csftdf_sf_file_not_found_error:
3593 0000B426 B802000000 <1>      mov      eax, 2 ; File not found
3594                  <1>      ; jmp      short csftdf_sf_error_retn
3595                  <1>      ; 25/08/2024
3596 0000B42B C3         <1>      retn
3597                  <1>
3598                  <1> csftdf_check_sf_cdrv:
3599 0000B42C 8A3D[62770100] <1>      mov      bh, [Current_Drv]
3600                  <1>
3601 0000B432 883D[77820100] <1>      mov      [csftdf_cdrv], bh ; 23/03/2016
3602                  <1>
3603 0000B438 8A15[4C810100] <1>      mov      dl, [SourceFile_Drv]
3604 0000B43E 38FA        <1>      cmp      dl, bh ; byte [Current_Drv]
3605 0000B440 7407        <1>      je       short csftdf_sf_check_directory
3606                  <1>
3607 0000B442 E899C2FFFF <1>      call     change_current_drive
3608 0000B447 72DA        <1>      jc       short csftdf_sf_error_retn
3609                  <1>
3610                  <1> csftdf_sf_check_directory:
3611 0000B449 BE[4D810100] <1>      mov      esi, SourceFile_Directory
3612 0000B44E 803E20      <1>      cmp      byte [esi], 20h
3613 0000B451 760F        <1>      jna      short csftdf_find_sf
3614                  <1>
3615                  <1> csftdf_sf_change_directory:
3616 0000B453 FE05[7E2E0100] <1>      inc      byte [Restore_CDIRE]
3617 0000B459 30E4        <1>      xor      ah, ah ; CD_COMMAND sign -> 0
3618 0000B45B E87BEFFFFF <1>      call     change_current_directory
3619 0000B460 72C1        <1>      jc       short csftdf_sf_error_retn
3620                  <1>
3621                  <1> ;csftdf_sf_change_prompt_dir_string:
3622                  <1> ;      call     change_prompt_dir_string
3623                  <1>
3624                  <1> csftdf_find_sf:
3625 0000B462 BE[8E810100] <1>      mov      esi, SourceFile_Name
3626 0000B467 66B80018    <1>      mov      ax, 1800h ; Except volume label and dirs
3627 0000B46B E8FAD4FFFF <1>      call     find_first_file
3628 0000B470 72B1        <1>      jc       short csftdf_sf_error_retn
3629                  <1>
3630                  <1> csftdf_sf_ambgfn_check:
3631 0000B472 6621D2      <1>      and      dx, dx ; Ambiguous filename chars used sign (DX>0)
3632 0000B475 7406        <1>      jz       short csftdf_sf_found
3633                  <1>
3634                  <1> csftdf_ambiguous_file_name_error:
3635                  <1> ;mov      eax, 2 ; File not found error
3636                  <1> ; 29/07/2022
3637 0000B477 29C0        <1>      sub      eax, eax
3638 0000B479 B002        <1>      mov      al, 2
3639 0000B47B F9         <1>      stc
3640 0000B47C C3         <1>      retn
3641                  <1>
3642                  <1> csftdf_sf_found:
3643 0000B47D A3[78820100] <1>      mov      [csftdf_filesize], eax
3644                  <1>
3645 0000B482 09C0        <1>      or       eax, eax
3646 0000B484 7504        <1>      jnz      short csftdf_set_source_file_direntry
3647                  <1>
3648                  <1> ;csftdf_sf_file_size_zero:
3649                  <1> ;mov      eax, 20 ; TRDOS zero length (file size) error
3650                  <1> ; 25/08/2024
3651                  <1> ; eax = 0
3652                  <1> ; 29/07/2022
3653                  <1> ;sub      eax, eax
3654 0000B486 B014        <1>      mov      al, 20
3655 0000B488 F9         <1>      stc
3656 0000B489 C3         <1>      retn
3657                  <1>
3658                  <1> csftdf_set_source_file_direntry:
3659 0000B48A BE[54800100] <1>      mov      esi, FindFile_DirEntry
3660 0000B48F BF[9E810100] <1>      mov      edi, SourceFile_DirEntry
3661                  <1> ;mov      ecx, 8
3662                  <1> ; 29/07/2022
3663 0000B494 31C9        <1>      xor      ecx, ecx
3664 0000B496 B108        <1>      mov      cl, 8
3665 0000B498 F3A5        <1>      rep      movsd
3666                  <1>
3667                  <1> csftdf_sf_restore_cdrv:
3668                  <1> ; 22/03/2016
3669 0000B49A 8A15[77820100] <1>      mov      dl, [csftdf_cdrv]
3670 0000B4A0 3A15[62770100] <1>      cmp      dl, [Current_Drv]
3671 0000B4A6 7407        <1>      je       short csftdf_sf_restore_cdir
3672 0000B4A8 E833C2FFFF <1>      call     change_current_drive
3673 0000B4AD 724F        <1>      jc       short csftdf_df_error_retn ; 30/03/2016
3674                  <1>
3675                  <1> csftdf_sf_restore_cdir:
3676 0000B4AF 803D[7E2E0100]00 <1>      cmp      byte [Restore_CDIRE], 0
3677 0000B4B6 7612        <1>      jna      short csftdf_df_check_filename_exists
3678 0000B4B8 29C0        <1>      sub      eax, eax
3679 0000B4BA BE00010900 <1>      mov      esi, Logical_DOSDisks
3680 0000B4BF 88D4        <1>      mov      ah, dl ; byte [csftdf_cdrv]
3681 0000B4C1 01C6        <1>      add      esi, eax
3682 0000B4C3 E8C9C2FFFF <1>      call     restore_current_directory
3683 0000B4C8 7234        <1>      jc       short csftdf_df_error_retn
3684                  <1>
3685                  <1> csftdf_df_check_filename_exists:
3686 0000B4CA 803D[0E820100]20 <1>      cmp      byte [DestinationFile_Name], 20h
3687 0000B4D1 7716        <1>      ja       short csftdf_check_df_cdrv
3688                  <1>
3689                  <1> csftdf_copy_sf_name:
3690 0000B4D3 BF[0E820100] <1>      mov      edi, DestinationFile_Name
3691 0000B4D8 BE[8E810100] <1>      mov      esi, SourceFile_Name
3692 0000B4DD B10C        <1>      mov      cl, 12
3693                  <1>
3694                  <1> csftdf_df_copy_sf_name_loop:
3695 0000B4DF AC         <1>      lodsb
3696 0000B4E0 AA         <1>      stosb
3697 0000B4E1 08C0        <1>      or       al, al
3698 0000B4E3 7404        <1>      jz       short csftdf_check_df_cdrv
3699 0000B4E5 FEC9        <1>      dec      cl
3700 0000B4E7 75F6        <1>      jnz      csftdf_df_copy_sf_name_loop
3701                  <1>
3702                  <1> csftdf_check_df_cdrv:

```

```

3703 0000B4E9 8A15[CC810100] <1> mov dl, [DestinationFile_Drv]
3704 0000B4EF 3A15[62770100] <1> cmp dl, [Current_Drv]
3705 0000B4F5 7408 <1> je short csftdf_df_check_directory
3706 <1>
3707 0000B4F7 E8E4C1FFFF <1> call change_current_drive
3708 0000B4FC 7301 <1> jnc short csftdf_df_check_directory
3709 <1>
3710 <1> csftdf_df_error_retn:
3711 0000B4FE C3 <1> retn
3712 <1>
3713 <1> csftdf_df_check_directory:
3714 0000B4FF BE[CD810100] <1> mov esi, DestinationFile_Directory
3715 0000B504 803E20 <1> cmp byte [esi], 20h
3716 0000B507 760F <1> jna short csftdf_df_find_df
3717 <1>
3718 <1> csftdf_df_change_directory:
3719 0000B509 FE05[7E2E0100] <1> inc byte [Restore_CDIRE]
3720 0000B50F 28E4 <1> sub ah, ah ; CD_COMMAND sign -> 0
3721 0000B511 E8C5EEFFFF <1> call change_current_directory
3722 0000B516 72E6 <1> jc short csftdf_df_error_retn
3723 <1>
3724 <1> ;csftdf_df_change_prompt_dir_string:
3725 <1> ; call change_prompt_dir_string
3726 <1>
3727 <1> csftdf_df_find_df:
3728 <1> ; 23/03/2016
3729 0000B518 29DB <1> sub ebx, ebx
3730 0000B51A 8A3D[CC810100] <1> mov bh, [DestinationFile_Drv]
3731 0000B520 81C300010900 <1> add ebx, Logical_DOSDisks
3732 0000B526 891D[A4820100] <1> mov [csftdf_df_drv_dt], ebx
3733 <1>
3734 0000B52C BE[0E820100] <1> mov esi, DestinationFile_Name
3735 <1> ;xor ax, ax
3736 <1> ; 25/08/2024
3737 0000B531 31C0 <1> xor eax, eax
3738 <1> ; DestinationFile_AttributesMask -> any/zero
3739 0000B533 E832D4FFFF <1> call find_first_file
3740 0000B538 7217 <1> jc short csftdf_df_check_error_code
3741 <1>
3742 <1> csftdf_df_ambgfn_check:
3743 0000B53A 6609D2 <1> or dx, dx ; Ambiguous filename chars used sign (DX>0)
3744 0000B53D 7529 <1> jnz short csftdf_df_error_inv_fname
3745 <1>
3746 <1> csftdf_df_found:
3747 0000B53F C605[76820100]01 <1> mov byte [DestinationFileFound], 1
3748 <1> ; 17/10/2016 (c1 -> b1)
3749 0000B546 80E31F <1> and bl, 1Fh ; Attributes, D-V-S-H-R
3750 0000B549 7452 <1> jz short csftdf_df_save_first_cluster
3751 <1>
3752 <1> csftdf_df_permission_denied_retn:
3753 <1> ;mov eax, 05h ; Access/Permission denied.
3754 <1> ; 29/07/2022
3755 0000B54B 29C0 <1> sub eax, eax
3756 0000B54D B005 <1> mov al, 5
3757 <1> csftdf_df_error_stc_retn:
3758 0000B54F F9 <1> stc
3759 0000B550 C3 <1> retn
3760 <1>
3761 <1> csftdf_df_check_error_code:
3762 <1> ;cmp eax, 2
3763 0000B551 3C02 <1> cmp al, 2
3764 0000B553 75FA <1> jne short csftdf_df_error_stc_retn
3765 <1>
3766 0000B555 C605[76820100]00 <1> mov byte [DestinationFileFound], 0
3767 <1>
3768 <1> ; 15/10/2016
3769 0000B55C BE[44800100] <1> mov esi, FindFile_Name ; *
3770 0000B561 E863D7FFFF <1> call check_filename
3771 0000B566 7306 <1> jnc short csftdf_df_valid_fname
3772 <1> csftdf_df_error_inv_fname: ; 'invalid file name !'
3773 <1> ;mov eax, ERR_INV_FILE_NAME ; 26
3774 <1> ; 29/07/2022
3775 0000B568 29C0 <1> sub eax, eax
3776 0000B56A B01A <1> mov al, ERR_INV_FILE_NAME ; 26
3777 0000B56C F9 <1> stc
3778 0000B56D C3 <1> retn
3779 <1>
3780 <1> csftdf_df_valid_fname:
3781 <1> ; 21/03/2016
3782 <1> ; (Capitalized file name)
3783 <1> ;mov esi, FindFile_Name ; * ; 15/10/2016
3784 0000B56E BF[0E820100] <1> mov edi, DestinationFile_Name
3785 0000B573 A5 <1> movsd
3786 0000B574 A5 <1> movsd
3787 0000B575 A5 <1> movsd
3788 <1> ;movsb
3789 <1>
3790 <1> csftdf_check_disk_free_size_0:
3791 0000B576 A1[BA810100] <1> mov eax, [SourceFile_DirEntry+DirEntry_FileSize]
3792 <1>
3793 <1> csftdf_check_disk_free_size_1:
3794 <1> ;sub ebx, ebx
3795 <1> ;mov esi, Logical_DOSDisks
3796 <1> ;mov bh, [DestinationFile_Drv]
3797 <1> ;add esi, ebx
3798 <1>
3799 0000B57B 8B35[A4820100] <1> mov esi, [csftdf_df_drv_dt] ; 23/03/2016
3800 <1>
3801 <1> ;movzx ecx, word [esi+LD_BPB+BytesPerSec] ; 17, LD_BPB + 0Bh
3802 <1> ; 19/12/2025
3803 0000B581 B900020000 <1> mov ecx, 512
3804 0000B586 01C8 <1> add eax, ecx
3805 0000B588 48 <1> dec eax ; file size (additional bytes) + 511 (round up)
3806 <1> csftdf_check_disk_free_size_3: ; 16/03/2016
3807 0000B589 29D2 <1> sub edx, edx
3808 0000B58B F7F1 <1> div ecx ; bytes per sector
3809 <1>
3810 <1> csftdf_check_disk_free_size:
3811 0000B58D 3B4674 <1> cmp eax, [esi+LD_FreeSectors]
3812 <1> ;jb csftdf_check_disk_free_size_ok
3813 <1> ; 25/08/2024
3814 <1> ; 29/07/2022
3815 <1> ;jb short csftdf_check_dfs_ok
3816 0000B590 7705 <1> ja short csftdf_df_insufficient_disk_space
3817 <1>
3818 <1> ; 25/08/2024
3819 <1> ;cmp byte [esi+LD_FATType], 0 ; FS needs FDT sector also.
3820 <1> ;ja csftdf_check_disk_free_size_ok
3821 <1> ; 29/07/2022
3822 <1> ;jna short csftdf_df_insufficient_disk_space
3823 <1>
3824 <1> csftdf_check_dfs_ok:
3825 0000B592 E985000000 <1> jmp csftdf_check_disk_free_size_ok
3826 <1>

```

```

3827 <1> csftdf_df_insufficient_disk_space:
3828 <1> ;mov eax, 27h ; insufficient disk space
3829 <1> ; 29/07/2022
3830 0000B597 29C0 <1> sub eax, eax
3831 0000B599 B027 <1> mov al, 27h
3832 <1> ;jmp short csftdf_df_error_stc_retn
3833 <1> ; 25/08/2024
3834 0000B59B F9 <1> stc
3835 0000B59C C3 <1> retn
3836 <1>
3837 <1> csftdf_df_save_first_cluster:
3838 <1> ; ESI = FindFile_DirEntry (for the old destination file)
3839 <1> ; EAX = Old destination file size
3840 <1> ; 24/03/2016
3841 <1> ; EDI = Directory entry address (within Dir Buffer boundaries)
3842 0000B59D 81EF00000800 <1> sub edi, Directory_Buffer ; (<65536)
3843 <1> ;shr di, 5 ; Convert entry offset to entry index/number
3844 <1> ; 29/07/2022
3845 0000B5A3 C1EF05 <1> shr edi, 5
3846 0000B5A6 66893D[46820100] <1> mov [DestinationFile_DirEntryNumber], di ; (<2048)
3847 <1>
3848 <1> csftdf_df_check_sf_df_fcluster:
3849 0000B5AD 668B5614 <1> mov dx, [esi+DirEntry_FstClusHI]
3850 0000B5B1 C1E210 <1> shl edx, 16
3851 0000B5B4 668B561A <1> mov dx, [esi+DirEntry_FstClusLO]
3852 0000B5B8 8915[88820100] <1> mov [csftdf_df_cluster], edx
3853 <1> csftdf_df_check_sf_df_fcluster_1:
3854 0000B5BE 668B15[B2810100] <1> mov dx, [SourceFile_DirEntry+DirEntry_FstClusHI]
3855 0000B5C5 C1E210 <1> shl edx, 16
3856 0000B5C8 668B15[B8810100] <1> mov dx, [SourceFile_DirEntry+DirEntry_FstClusLO]
3857 0000B5CF 3B15[88820100] <1> cmp edx, [csftdf_df_cluster]
3858 0000B5D5 7512 <1> jne short csftdf_df_check_sf_df_fcluster_ok
3859 <1> csftdf_df_check_sf_df_drv:
3860 0000B5D7 8A15[4C810100] <1> mov dl, [SourceFile_Drv]
3861 0000B5DD 3A15[CC810100] <1> cmp dl, [DestinationFile_Drv]
3862 0000B5E3 7504 <1> jne short csftdf_df_check_sf_df_fcluster_ok
3863 <1>
3864 <1> ; source and destination files are same !
3865 <1> ; (they have same first cluster value on same logical disk)
3866 <1>
3867 0000B5E5 31C0 <1> xor eax, eax ; mov eax, 0 -> Bad command or file name !
3868 0000B5E7 F9 <1> stc
3869 0000B5E8 C3 <1> retn
3870 <1>
3871 <1> csftdf_df_check_sf_df_fcluster_ok:
3872 <1> csftdf_df_move_findFile_struct:
3873 <1> ; mov esi, FindFile_DirEntry
3874 0000B5E9 BF[1E820100] <1> mov edi, DestinationFile_DirEntry
3875 <1> ;mov ecx, 8
3876 0000B5EE 31C9 <1> xor ecx, ecx
3877 0000B5F0 B108 <1> mov cl, 8
3878 0000B5F2 F3A5 <1> rep movsd
3879 <1>
3880 <1> ;csftdf_check_disk_free_size_2:
3881 0000B5F4 89C2 <1> mov edx, eax ; Old destination file size
3882 <1>
3883 <1> ;mov eax, [SourceFile_DirEntry+DirEntry_FileSize]
3884 0000B5F6 A1[78820100] <1> mov eax, [csftdf_filesize] ; 23/03/2016
3885 <1>
3886 <1> ;sub ecx, ecx ; 0
3887 <1> ;mov esi, Logical_DOSDisks
3888 <1> ;mov ch, [DestinationFile_Drv]
3889 <1> ;add esi, ecx
3890 <1> ;
3891 <1> ;mov [csftdf_df_drv_dt], esi
3892 <1>
3893 0000B5FB 8B35[A4820100] <1> mov esi, [csftdf_df_drv_dt] ; 23/03/2016
3894 <1>
3895 <1> ;mov cx, [esi+LD_BPB+BytesPerSec] ; 17, LD_BPB + 0Bh
3896 <1> ; 19/12/2025
3897 0000B601 66B90002 <1> mov cx, 512
3898 0000B605 01CA <1> add edx, ecx ; + 512
3899 0000B607 01C8 <1> add eax, ecx ; + 512
3900 0000B609 4A <1> dec edx ; old file size + 511 (round up)
3901 0000B60A 48 <1> dec eax ; new file size + 511 (round up)
3902 0000B60B F7D9 <1> neg ecx ; -512 ; 0FFFFFFE00h
3903 0000B60D 21CA <1> and edx, ecx ; = old sector count * 512
3904 0000B60F 21C8 <1> and eax, ecx ; = new sector count * 512
3905 <1>
3906 0000B611 29D0 <1> sub eax, edx ; new file size - old file size (on disk)
3907 0000B613 7607 <1> jna short csftdf_check_disk_free_size_ok
3908 <1>
3909 0000B615 F7D9 <1> neg ecx ; 512 (bytes per sector) ; 200h
3910 <1> ; check free space for additional sectors
3911 <1> ; eax = number of additional sectors * bytes per sector
3912 <1> ; esi = Logical DOS drive number (of destination disk)
3913 0000B617 E96DFFFFFF <1> jmp csftdf_check_disk_free_size_3
3914 <1>
3915 <1> csftdf_check_disk_free_size_ok:
3916 <1> ; 18/03/2016
3917 <1> csftdf_df_check_copy_cmd_phase:
3918 0000B61C A0[74820100] <1> mov al, [copy_cmd_phase]
3919 0000B621 3C01 <1> cmp al, 1
3920 0000B623 7514 <1> jne short csftdf2_check_cdrv
3921 <1>
3922 0000B625 31C0 <1> xor eax, eax
3923 0000B627 A2[74820100] <1> mov [copy_cmd_phase], al ; 0
3924 <1>
3925 0000B62C 8A15[76820100] <1> mov dl, [DestinationFileFound]
3926 0000B632 8A35[A9810100] <1> mov dh, [SourceFile_DirEntry+11] ; Attributes
3927 <1>
3928 <1> csftdf_return:
3929 0000B638 C3 <1> retn
3930 <1>
3931 <1> ; Phase 2
3932 <1> csftdf_ph2:
3933 <1> ; 29/07/2022
3934 <1> csftdf2_check_cdrv:
3935 <1> ; 18/03/2016
3936 <1> ; Here, destination drive and directory are ready !
3937 <1> ; (checking/restoring is not needed)
3938 <1> ; (Since at the end of the phase 1)
3939 <1>
3940 <1> ; mov dl, [DestinationFile_Drv]
3941 <1> ; cmp dl, [Current_Drv]
3942 <1> ; je short csftdf2_df_check_directory
3943 <1> ;
3944 <1> ; call change_current_drive
3945 <1> ; jc short csftdf2_read_error
3946 <1> ;
3947 <1> ;csftdf2_df_check_directory:
3948 <1> ; mov esi, DestinationFile_Directory
3949 <1> ; cmp byte [esi], 20h
3950 <1> ; jna short csftdf2_df_check_found_or_not

```

```

3951 <1> ;
3952 <1> ;csftdf2_df_change_directory:
3953 <1> ; inc byte [Restore_CDIR]
3954 <1> ; xor ah, ah ; CD_COMMAND sign -> 0
3955 <1> ; call change_current_directory
3956 <1> ; jc short csftdf2_stc_return
3957 <1> ;
3958 <1> ;csftdf2_df_change_prompt_dir_string:
3959 <1> ; call change_prompt_dir_string
3960 <1> ;
3961 <1> csftdf2_df_check_found_or_not:
3962 <1> ; 21/03/2016
3963 0000B639 803D[76820100]00 <1> cmp byte [DestinationFileFound], 0
3964 0000B640 7742 <1> ja short csftdf2_set_sf_percentage
3965 <1> ;
3966 <1> csftdf2_create_file:
3967 0000B642 BE[0E820100] <1> mov esi, DestinationFile_Name
3968 0000B647 A1[78820100] <1> mov eax, [csftdf_filesize]
3969 0000B64C 30C9 <1> xor cl, cl ; 0
3970 <1> ;
3971 0000B64E 31DB <1> xor ebx, ebx ; 0
3972 0000B650 4B <1> dec ebx ; 0FFFFFFFh
3973 <1> ;
3974 <1> ; INPUT ->
3975 <1> ; EAX -> File Size
3976 <1> ; ESI = ASCIIZ File name
3977 <1> ; CL = File attributes
3978 <1> ; EBX = FFFFFFFFh -> empty file sign for FAT fs
3979 <1> ; EBX <> FFFFFFFFh -> use file size for FAT fs
3980 <1> ;
3981 <1> ; OUTPUT ->
3982 <1> ; EAX = New file's first cluster
3983 <1> ; (0 for empty file) ; 29/08/2024
3984 <1> ; ESI = Logical Dos Drv Descr. Table Addr.
3985 <1> ; EBX = CreateFile_Size address
3986 <1> ; ECX = Sectors per cluster (<256)
3987 <1> ; EDX = Directory Entry Index/Number (<65536)
3988 <1> ; 29/08/2024
3989 <1> ; EBX = File Size (0 for a new, empty file)
3990 <1> ; ECX = Directory Entry Index/Number (<2048)
3991 <1> ; (in directory cluster, not in directory)
3992 <1> ; EDX = Directory Cluster Number (of the file)
3993 <1> ;
3994 <1> ; cf = 1 -> error code in AL (EAX)
3995 <1> ;
3996 0000B651 E80D060000 <1> call create_file
3997 <1> ;pop esi
3998 <1> ;jc csftdf2_rw_error
3999 <1> ; 29/07/2022
4000 0000B656 7305 <1> jnc short csftdf2_create_file_OK
4001 0000B658 E978050000 <1> jmp csftdf2_rw_error
4002 <1> ;
4003 <1> csftdf2_create_file_OK:
4004 <1> ;
4005 0000B65D A3[88820100] <1> mov [csftdf_df_cluster], eax
4006 <1> ; 27/08/2024
4007 <1> ; eax = 0
4008 <1> ;
4009 <1> ; 24/03/2016
4010 <1> ;mov [DestinationFile_DirEntryNumber], dx
4011 <1> ; 29/08/2024
4012 <1> ; (these 3 parameters are needed for
4013 <1> ; loading same dir entry for first cluster update)
4014 0000B662 8915[42820100] <1> mov [DestinationFile_DirCluster], edx
4015 0000B668 66890D[46820100] <1> mov [DestinationFile_DirEntryNumber], cx
4016 <1> ;
4017 <1> ; 30/08/2024
4018 <1> ; edx -> ecx
4019 <1> ; 21/03/2016
4020 0000B66F BE00000800 <1> mov esi, Directory_Buffer
4021 <1> ;shl edx, 5 ; 32 * index number
4022 0000B674 C1E105 <1> shl ecx, 5
4023 <1> ;add esi, edx
4024 0000B677 01CE <1> add esi, ecx
4025 0000B679 BF[1E820100] <1> mov edi, DestinationFile_DirEntry
4026 <1> ;mov cl, 8 ; 32 bytes
4027 <1> ; 29/08/2024
4028 0000B67E 29C9 <1> sub ecx, ecx
4029 0000B680 B108 <1> mov cl, 8
4030 0000B682 F3A5 <1> rep movsd
4031 <1> ;
4032 <1> ;mov cl, [esi] ; L.D.D.D.T.
4033 <1> ;mov [DestinationFile_Drv], cl
4034 <1> ;
4035 <1> csftdf2_set_sf_percentage:
4036 <1> ; 17/03/2016
4037 0000B684 31C0 <1> xor eax, eax
4038 0000B686 A2[9C820100] <1> mov [csftdf_percentage], al ; 0, reset
4039 <1> ;
4040 0000B68B A3[94820100] <1> mov [csftdf_sf_rbytes], eax ; 0, reset
4041 0000B690 A3[98820100] <1> mov [csftdf_df_wbytes], eax ; 0, reset
4042 <1> ;
4043 0000B695 8A25[4C810100] <1> mov ah, [SourceFile_Drv]
4044 0000B69B BE00010900 <1> mov esi, Logical_DOSDisks
4045 0000B6A0 01C6 <1> add esi, eax
4046 <1> ;
4047 0000B6A2 8935[A0820100] <1> mov [csftdf_sf_drv_dt], esi ; 23/03/2016
4048 <1> ;
4049 0000B6A8 668B15[B2810100] <1> mov dx, [SourceFile_DirEntry+DirEntry_FstClusHI]
4050 0000B6AF C1E210 <1> shl edx, 16
4051 0000B6B2 668B15[B8810100] <1> mov dx, [SourceFile_DirEntry+DirEntry_FstClusLO]
4052 0000B6B9 8915[84820100] <1> mov [csftdf_sf_cluster], edx
4053 <1> ;
4054 <1> ; 16/03/2016
4055 <1> ; Note: Singlix FS boot sector parameters (for cluster
4056 <1> ; related calculations) has same offset
4057 <1> ; values from LD_BPB as in FAT file system.
4058 <1> ; [esi+LD_BPB+SecPerClust] is 1 for Singlix FS.
4059 <1> ;
4060 0000B6BF 0FB64E13 <1> movzx ecx, byte [esi+LD_BPB+SecPerClust]
4061 0000B6C3 880D[CA810100] <1> mov [SourceFile_SecPerClust], cl
4062 <1> ;
4063 <1> ; 19/12/2025
4064 <1> %if 0
4065 <1> ; 17/03/2016
4066 <1> cmp [esi+LD_FATType], ch ; 0
4067 <1> ja short csftdf2_set_sf_percent_rsize1
4068 <1> ;
4069 <1> mov eax, 65536 ; read/write buffer size for Singlix FS
4070 <1> jmp short csftdf2_set_sf_percent_rsize2
4071 <1> %endif
4072 <1> ;
4073 <1> csftdf2_set_sf_percent_rsize1:
4074 <1> ;mov ax, [esi+LD_BPB+BytesPerSec]

```



```

4075      <1>      ; 19/12/2025
4076 0000B6C9 66B80002      <1>      mov     ax, 512
4077 0000B6CD F7E1          <1>      mul     ecx
4078                                <1>      ;sub     edx, edx
4079                                <1>  csftdf2_set_sf_percent_rsize2:
4080 0000B6CF A3[8C820100]    <1>      mov     [csftdf_r_size], eax
4081                                <1>
4082                                <1>  csftdf2_set_df_percentage:
4083                                <1>      ;sub     eax, eax
4084                                <1>      ;mov     ah, [DestinationFile_Drv]
4085                                <1>      ;mov     edi, Logical_DOSDisks
4086                                <1>      ;add     edi, eax
4087                                <1>      ;mov     [csftdf_df_drv_dt], edi ; 17/03/2016
4088                                <1>
4089 0000B6D4 8B3D[A4820100]    <1>      mov     edi, [csftdf_df_drv_dt] ; 23/03/2016
4090                                <1>
4091                                <1>      ; 16/03/2016
4092                                <1>      ; Note: Singlix FS boot sector parameters (for cluster
4093                                <1>      ; related calculations) has same offset
4094                                <1>      ; values from LD_BPB as in FAT file system.
4095                                <1>      ; [edi+LD_BPB+SecPerClust] is 1 for Singlix FS.
4096                                <1>      ;
4097                                <1>      ;movzx   ecx, byte [edi+LD_BPB+SecPerClust]
4098 0000B6DA 8A4F13          <1>      mov     cl, [edi+LD_BPB+SecPerClust]
4099 0000B6DD 880D[4A820100]    <1>      mov     [DestinationFile_SecPerClust], cl
4100                                <1>
4101                                <1>      ; 19/12/2025
4102                                <1>      %if 0
4103                                <1>      ; 17/03/2016
4104                                <1>      cmp     [edi+LD_FATType], ch ; 0
4105                                <1>      ja      short csftdf2_set_df_percent_wsize1
4106                                <1>
4107                                <1>      mov     eax, 65536 ; read/write buffer size for Singlix FS
4108                                <1>      jmp     short csftdf2_set_df_percent_wsize2
4109                                <1>      %endif
4110                                <1>
4111                                <1>  csftdf2_set_df_percent_wsize1:
4112                                <1>      ;movzx   eax, word [edi+LD_BPB+BytesPerSec]
4113                                <1>      ; 19/12/2025
4114 0000B6E3 B800020000      <1>      mov     eax, 512
4115 0000B6E8 F7E1          <1>      mul     ecx
4116                                <1>      ;sub     edx, edx
4117                                <1>  csftdf2_set_df_percent_wsize2:
4118 0000B6EA A3[90820100]    <1>      mov     [csftdf_w_size], eax
4119                                <1>
4120 0000B6EF A1[78820100]    <1>      mov     eax, [csftdf_filesize]
4121                                <1>
4122 0000B6F4 3D00000100      <1>      cmp     eax, 65536 ; 64KB ; small file
4123 0000B6F9 721F          <1>      jnb     short csftdf2_load_file ; do not display percentage
4124                                <1>
4125                                <1>  csftdf2_reset_wf_percent_ptr_chk_64k:
4126 0000B6FB B201          <1>      mov     dl, 1 ; 25/03/2016
4127                                <1>
4128 0000B6FD 3D00000400      <1>      cmp     eax, 65536*4 ; 256KB
4129 0000B702 7310          <1>      jnb     short csftdf2_enable_percentage_display ; big file
4130                                <1>
4131                                <1>      ; 64-128KB file size for floppy disks
4132 0000B704 3815[4C810100]    <1>      cmp     byte [SourceFile_Drv], dl ; 1 ; read from floppy disk ?
4133 0000B70A 7608          <1>      jna     short csftdf2_enable_percentage_display
4134                                <1>
4135 0000B70C 3815[CC810100]    <1>      cmp     byte [DestinationFile_Drv], dl ; 1 ; write to floppy disk ?
4136 0000B712 7706          <1>      ja      short csftdf2_load_file
4137                                <1>
4138                                <1>  csftdf2_enable_percentage_display:
4139 0000B714 8815[9C820100]    <1>      mov     [csftdf_percentage], dl ; 1
4140                                <1>
4141                                <1>  csftdf2_load_file:
4142                                <1>      ; 13/05/2016
4143                                <1>      ; 19/03/2016
4144                                <1>      ; 18/03/2016
4145                                <1>      ; 17/03/2016
4146 0000B71A B40F          <1>      mov     ah, 0Fh
4147 0000B71C E80360FFFF      <1>      call    _int10h
4148                                <1>      ; 13/05/2016
4149 0000B721 883D[9D820100]    <1>      mov     [csftdf_videopage], bh ; active video page
4150 0000B727 B403          <1>      mov     ah, 03h
4151 0000B729 E8F65FFFFF      <1>      call    _int10h
4152 0000B72E 668915[9E820100]    <1>      mov     [csftdf_cursorpos], dx
4153                                <1>
4154 0000B735 29C0          <1>      sub     eax, eax
4155 0000B737 A2[75820100]    <1>      mov     [csftdf_rw_err], al ; 0
4156                                <1>
4157                                <1>      ; ///
4158                                <1>  csftdf_sf_amb: ; 15/03/2016
4159 0000B73C 8B0D[78820100]    <1>      mov     ecx, [csftdf_filesize]; 23/03/2016
4160                                <1>
4161                                <1>      ; TRDOS 386 (TRDOS v2.0)
4162                                <1>      ; Allocate contiguous memory block for loading the file
4163                                <1>
4164                                <1>      ;mov     ecx, [SourceFile_DirEntry+DirEntry_FileSize]
4165                                <1>
4166                                <1>      ;sub     eax, eax ; First free memory aperture
4167                                <1>
4168                                <1>      ; eax = 0 (Allocate memory from the beginning)
4169                                <1>      ; ecx = File (Allocation) size in bytes
4170                                <1>
4171                                <1>      ; 31/08/2024 - temporary
4172                                <1>      %if 0
4173 0000B742 E8F2A5FFFF      <1>      call    allocate_memory_block
4174 0000B747 7304          <1>      jnc     short loc_check_sf_save_loading_parms
4175                                <1>      %endif
4176                                <1>
4177                                <1>  csftdf_use_cluster_buff: ; 27/08/2024
4178                                <1>
4179 0000B749 29C0          <1>      sub     eax, eax
4180 0000B74B 29C9          <1>      sub     ecx, ecx
4181                                <1>
4182                                <1>  loc_check_sf_save_loading_parms:
4183 0000B74D A3[7C820100]    <1>      mov     [csftdf_sf_mem_addr], eax ; loading address
4184 0000B752 890D[80820100]    <1>      mov     [csftdf_sf_mem_bsize], ecx ; block size
4185                                <1>      ; ///
4186                                <1>      ; 19/03/2016
4187 0000B758 8B35[A0820100]    <1>      mov     esi, [csftdf_sf_drv_dt] ; logical dos drv desc. tbl.
4188                                <1>
4189                                <1>      ; 17/03/2016
4190 0000B75E 09C0          <1>      or      eax, eax ; contiguous free memory block address
4191                                <1>      ;jz      csftdf2_read_sf_cluster
4192                                <1>      ; 29/07/2022
4193 0000B760 7505          <1>      jnz     short csftdf2_1
4194 0000B762 E949010000      <1>      jmp     csftdf2_read_sf_cluster
4195                                <1>
4196                                <1>  csftdf2_1:
4197                                <1>      ; 18/03/2016
4198 0000B767 8B1D[7C820100]    <1>      mov     ebx, [csftdf_sf_mem_addr] ; memory block address

```

```

4199 <1>
4200 <1> ; 19/12/2025
4201 <1> %if 0
4202 <1>     cmp     byte [esi+LD_FATType], 0
4203 <1>             ;jna     csftdf2_load_fs_file
4204 <1>             ; 29/07/2022
4205 <1>     ja      short csftdf2_load_fat_file
4206 <1>     jmp     csftdf2_load_fs_file
4207 <1> %endif
4208 <1>
4209 <1> csftdf2_load_fat_file:
4210 <1>     push    ebx ; *
4211 <1>
4212 <1> csftdf2_load_fat_file_next:
4213 <1>     mov     esi, msg_reading
4214 <1>     call    print_msg
4215 <1>
4216 <1>     cmp     byte [csftdf_percentage], 0
4217 <1>     jna     short csftdf2_load_fat_file_1
4218 <1>
4219 <1>     call    csftdf2_print_percentage ; 19/03/2016
4220 <1>
4221 <1> csftdf2_load_fat_file_1:
4222 <1>     mov     esi, [csftdf_sf_drv_dt]
4223 <1>     pop     ebx ; *
4224 <1>
4225 <1> csftdf2_load_fat_file_2:
4226 <1>     call    csftdf2_read_fat_file_sectors ; 19/03/2016
4227 <1>     ;jc     csftdf2_rw_error ; eocc! or disk error!
4228 <1>     ; 29/07/2022
4229 <1>     jnc     short csftdf2_load_fat_file_3
4230 <1>     jmp     csftdf2_rw_error
4231 <1> csftdf2_load_fat_file_3:
4232 <1>     or      edx, edx ; edx > 0 -> EOF
4233 <1>     jnz     short csftdf2_load_fat_file_ok
4234 <1>
4235 <1>     cmp     byte [csftdf_percentage], 0
4236 <1>     jna     short csftdf2_load_fat_file_2
4237 <1>
4238 <1>     push    ebx ; *
4239 <1>
4240 <1>     ; Set cursor position
4241 <1>     ; AH= 02h, BH= Page Number, DH= Row, DL= Column
4242 <1>     mov     bh, [csftdf_videopage]
4243 <1>     mov     dx, [csftdf_cursorpos]
4244 <1>     mov     ah, 2
4245 <1>     call    _int10h
4246 <1>     jmp     short csftdf2_load_fat_file_next
4247 <1>
4248 <1> csftdf2_load_fat_file_ok:
4249 <1>     cmp     byte [csftdf_percentage], 0
4250 <1>     ;jna     csftdf2_save_file ; 25/03/2016
4251 <1>     ; 29/07/2022
4252 <1>     ja      short csftdf2_2
4253 <1>     jmp     csftdf2_save_file
4254 <1> csftdf2_2:
4255 <1>     ; "Reading... 100%"
4256 <1>     mov     edi, percentagestr
4257 <1>     mov     al, '1'
4258 <1>     stosb
4259 <1>     mov     al, '0'
4260 <1>     stosb
4261 <1>     stosb
4262 <1>
4263 <1>     mov     bh, [csftdf_videopage]
4264 <1>     mov     dx, [csftdf_cursorpos]
4265 <1>     mov     ah, 2
4266 <1>     call    _int10h
4267 <1>
4268 <1>     mov     esi, msg_reading
4269 <1>     call    print_msg
4270 <1>
4271 <1>     mov     esi, percentagestr
4272 <1>     call    print_msg
4273 <1>
4274 <1>     jmp     csftdf2_save_file ; 25/03/2016
4275 <1>
4276 <1> csftdf2_print_percentage:
4277 <1>     ; 09/12/2017
4278 <1>     ; 19/03/2016
4279 <1>     ; 18/03/2016
4280 <1>     mov     al, 20h
4281 <1>     mov     edi, percentagestr
4282 <1>     stosb
4283 <1>     stosb
4284 <1>     mov     eax, [csftdf_sf_rbytes]
4285 <1>     ;mov     edx, 100
4286 <1>     ; 29/07/2022
4287 <1>     sub     edx, edx
4288 <1>     mov     dl, 100
4289 <1>     mul     edx
4290 <1>     mov     ecx, [csftdf_filesize]
4291 <1>     div     ecx
4292 <1>     mov     cl, 10
4293 <1>     div     cl
4294 <1>     add     ah, '0'
4295 <1>     mov     [edi], ah
4296 <1>     and     al, al
4297 <1>     jz      short csftdf2_print_percent_1
4298 <1>     dec     edi
4299 <1>     ;cbw
4300 <1>     sub     ah, ah ; 09/12/2017
4301 <1>     div     cl
4302 <1>     add     ah, '0'
4303 <1>     mov     [edi], ah
4304 <1>     ;and     al, al
4305 <1>     ;jz      short csftdf2_print_percent_1
4306 <1>     ;dec     edi
4307 <1>     ;mov     [edi], '1' ; 100%
4308 <1>
4309 <1> csftdf2_print_percent_1:
4310 <1>     mov     esi, percentagestr
4311 <1>     ;call    print_msg
4312 <1>     ;retn
4313 <1>     jmp     print_msg
4314 <1>
4315 <1> csftdf2_read_file_sectors:
4316 <1>
4317 <1> ; 19/12/2025
4318 <1> %if 0
4319 <1>     ; 19/03/2016
4320 <1>     cmp     byte [esi+LD_FATType], 0
4321 <1>     ;jna     csftdf2_read_fs_file_sectors
4322 <1>     ; 29/07/2022

```

```

4323     <1>     ja      short csftdf2_read_fat_file_sectors
4324     <1>     jmp     csftdf2_read_fs_file_sectors
4325     <1> %endif
4326     <1>
4327     <1> csftdf2_read_fat_file_sectors:
4328     <1>     ; 19/03/2016
4329     <1>     ; 18/03/2016
4330     <1>     ; return:
4331     <1>     ; CF = 0 & EDX > 0 -> END OF FILE
4332     <1>     ; CF = 0 & EDX = 0 -> not EOF
4333     <1>     ; CF = 1 -> read error (error code in AL)
4334     <1>
4335     <1> csftdf2_read_fat_file_secs_0:
4336     <1>     mov     edx, [csftdf_filesize]
4337     <1>     sub     edx, [csftdf_sf_rbytes]
4338     <1>     cmp     edx, [csftdf_r_size]
4339     <1>     jnb     short csftdf2_read_fat_file_secs_1
4340     <1>     mov     [csftdf_r_size], edx
4341     <1>
4342     <1> csftdf2_read_fat_file_secs_1:
4343     <1>     mov     eax, [csftdf_r_size]
4344     <1>     sub     edx, edx
4345     <1>     ;movzx ecx, word [esi+LD_BPB+BytesPerSec]
4346     <1>     ; 19/12/2025
4347     <1>     mov     ecx, 512
4348     <1>     add     eax, ecx
4349     <1>     dec     eax
4350     <1>     div     ecx
4351     <1>     mov     ecx, eax ; sector count
4352     <1>     mov     eax, [csftdf_sf_cluster]
4353     <1>
4354     <1>     ; EBX = memory block address (current)
4355     <1>
4356     <1>     call    read_fat_file_sectors
4357     <1>     jc     short csftdf2_read_fat_file_secs_3
4358     <1>
4359     <1>     ; EBX = next memory address
4360     <1>
4361     <1>     mov     eax, [csftdf_sf_rbytes]
4362     <1>     add     eax, [csftdf_r_size]
4363     <1>     mov     edx, [csftdf_filesize]
4364     <1>     cmp     eax, edx
4365     <1>     jnb     short csftdf2_read_fat_file_secs_3 ; edx > 0
4366     <1>     mov     [csftdf_sf_rbytes], eax
4367     <1>
4368     <1>     push    ebx ; *
4369     <1>     ; get next cluster (csftdf_r_size! bytes)
4370     <1>     mov     eax, [csftdf_sf_cluster]
4371     <1>     call    get_next_cluster
4372     <1>     pop     ebx ; *
4373     <1>     jnc     short csftdf2_read_fat_file_secs_2
4374     <1>
4375     <1>     ; 15/10/2016
4376     <1>     ; Disk read error instead of drv not ready err
4377     <1>     mov     eax, 17 ; Read error !
4378     <1>     retn
4379     <1>
4380     <1> csftdf2_read_fat_file_secs_2:
4381     <1>     sub     edx, edx ; 0
4382     <1>     mov     [csftdf_sf_cluster], eax ; next cluster
4383     <1>
4384     <1> csftdf2_read_fat_file_secs_3:
4385     <1>     retn
4386     <1>
4387     <1> csftdf2_read_sf_cluster:
4388     <1>     ; 19/03/2016
4389     <1>     mov     ebx, Cluster_Buffer ; buffer address (64KB)
4390     <1>
4391     <1>     cmp     byte [csftdf_percentage], 0
4392     <1>     jna     short csftdf2_read_sf_clust_2
4393     <1>
4394     <1>     push    ebx ; *
4395     <1>
4396     <1> csftdf2_read_sf_clust_next:
4397     <1>     call    csftdf2_print_percentage
4398     <1>
4399     <1> csftdf2_read_sf_clust_0:
4400     <1>     mov     esi, [csftdf_sf_drv_dt]
4401     <1> csftdf2_read_sf_clust_1:
4402     <1>     pop     ebx ; *
4403     <1>
4404     <1> csftdf2_read_sf_clust_2:
4405     <1>     mov     edx, ebx
4406     <1>     add     edx, [csftdf_r_size]
4407     <1>     cmp     edx, Cluster_Buffer + 65536
4408     <1>     ja     short csftdf2_write_df_cluster
4409     <1>
4410     <1>     call    csftdf2_read_file_sectors ; 19/03/2016
4411     <1>     ;jc     csftdf2_save_fat_file_err2 ; eocc! or disk error!
4412     <1>     ; 29/07/2022
4413     <1>     jc     short csftdf2_3
4414     <1>
4415     <1>     or     edx, edx ; edx > 0 -> EOF
4416     <1>     jnz     short csftdf2_write_df_cluster
4417     <1>
4418     <1>     cmp     byte [csftdf_percentage], 0
4419     <1>     jna     short csftdf2_read_sf_clust_2
4420     <1>
4421     <1>     push    ebx ; *
4422     <1>
4423     <1>     ; Set cursor position
4424     <1>     ; AH= 02h, BH= Page Number, DH= Row, DL= Column
4425     <1>     mov     bh, [csftdf_videopage]
4426     <1>     mov     dx, [csftdf_cursorpos]
4427     <1>     mov     ah, 2
4428     <1>     call    _int10h
4429     <1>     jmp     short csftdf2_read_sf_clust_next
4430     <1>
4431     <1> csftdf2_write_df_cluster:
4432     <1>
4433     <1>     ; 31/08/2024
4434     <1> %if 1
4435     <1>     ; 27/08/2024
4436     <1>     ; 19/03/2016
4437     <1>     ;mov     esi, [csftdf_df_drv_dt] ; (!)
4438     <1>     ;mov     ebx, Cluster_Buffer ; buffer address (64KB)
4439     <1>     ;;;
4440     <1>     ; 27/08/2024
4441     <1>     call    csftdf2_update_df_fclust
4442     <1>     ; 31/08/2024
4443     <1>     jnc     short csftdf2_update_df_fclust_ok
4444     <1>     jmp     csftdf2_rw_error
4445     <1>
4446     <1> csftdf2_update_df_fclust_ok:

```

```

4447 <1> %endif
4448 <1> ; esi = [csftdf_df_drv_dt]
4449 <1>
4450 <1> ;mov esi, [csftdf_df_drv_dt]
4451 0000B912 BB00000700 <1> mov ebx, Cluster_Buffer ; buffer address (64KB)
4452 <1>
4453 <1> csftdf2_write_df_clust_next:
4454 0000B917 E85B000000 <1> call csftdf2_write_file_sectors ; 19/03/2016
4455 <1> ;jc csftdf2_save_fat_file_err2 ; eocc! or disk error!
4456 <1> ; 29/07/2022
4457 0000B91C 7305 <1> jnc short csftdf2_4
4458 <1> csftdf2_3:
4459 0000B91E E939020000 <1> jmp csftdf2_save_fat_file_err2
4460 <1> csftdf2_4:
4461 0000B923 09D2 <1> or edx, edx ; edx > 0 -> EOF
4462 0000B925 7513 <1> jnz short csftdf2_rw_f_clust_ok
4463 <1>
4464 0000B927 81FB00000800 <1> cmp ebx, Cluster_Buffer + 65536
4465 0000B92D 72E8 <1> jb short csftdf2_write_df_clust_next
4466 <1>
4467 <1> ; 31/08/2024
4468 0000B92F 8B35[A0820100] <1> mov esi, [csftdf_sf_drv_dt]
4469 <1>
4470 0000B935 E976FFFFFF <1> jmp csftdf2_read_sf_cluster ; 31/08/2024
4471 <1>
4472 <1> csftdf2_rw_f_clust_ok:
4473 0000B93A 803D[9C820100]00 <1> cmp byte [csftdf_percentage], 0
4474 <1> ;jna csftdf2_save_fat_file_4 ; 25/03/2016
4475 <1> ; 29/07/2022
4476 0000B941 762A <1> jna short csftdf2_5
4477 <1>
4478 <1> ; "100%"
4479 0000B943 BF[EE340100] <1> mov edi, percentagestr
4480 0000B948 B031 <1> mov al, '1'
4481 0000B94A AA <1> stosb
4482 0000B94B B030 <1> mov al, '0'
4483 0000B94D AA <1> stosb
4484 0000B94E AA <1> stosb
4485 <1>
4486 0000B94F 8A3D[9D820100] <1> mov bh, [csftdf_videopage]
4487 0000B955 668B15[9E820100] <1> mov dx, [csftdf_cursorpos]
4488 0000B95C B402 <1> mov ah, 2
4489 0000B95E E8C15DFFFF <1> call _int10h
4490 <1>
4491 0000B963 BE[EE340100] <1> mov esi, percentagestr
4492 0000B968 E83DB4FFFF <1> call print_msg
4493 <1> csftdf2_5:
4494 0000B96D E974010000 <1> jmp csftdf2_save_fat_file_4
4495 <1>
4496 <1> csftdf2_load_fs_file:
4497 <1> ; temporary - 18/03/2016
4498 0000B972 E95C020000 <1> jmp csftdf2_read_error
4499 <1>
4500 <1> csftdf2_write_file_sectors:
4501 <1>
4502 <1> ; 19/12/2025
4503 <1> %if 0
4504 <1> ; 31/08/2024
4505 <1> ; 30/08/2024
4506 <1> ; 19/03/2016
4507 <1> cmp byte [esi+LD_FATType], 0
4508 <1> ;jna csftdf2_write_fs_file_sectors
4509 <1> ; 29/07/2022
4510 <1> ja short csftdf2_write_fat_file_sectors
4511 <1> jmp csftdf2_write_fs_file_sectors
4512 <1> %endif
4513 <1>
4514 <1> csftdf2_write_fat_file_sectors:
4515 <1> ; 30/08/2024
4516 <1> ; 19/03/2016
4517 <1> ; 18/03/2016
4518 <1> ; return:
4519 <1> ; CF = 0 & EDX > 0 -> END OF FILE
4520 <1> ; CF = 0 & EDX = 0 -> not EOF
4521 <1> ; CF = 1 -> write error (error code in AL)
4522 <1>
4523 <1> csftdf2_write_fat_file_secs_0:
4524 0000B977 8B15[78820100] <1> mov edx, [csftdf_filesizes]
4525 0000B97D 2B15[98820100] <1> sub edx, [csftdf_df_wbytes]
4526 0000B983 3B15[90820100] <1> cmp edx, [csftdf_w_size]
4527 0000B989 7306 <1> jnb short csftdf2_write_fat_file_secs_1
4528 0000B98B 8915[90820100] <1> mov [csftdf_w_size], edx
4529 <1>
4530 <1> csftdf2_write_fat_file_secs_1:
4531 <1>
4532 <1> ; 31/08/2024
4533 <1> %if 0
4534 <1> ;;;
4535 <1> ; 30/08/2024
4536 <1> mov eax, [csftdf_df_cluster]
4537 <1> and eax, eax
4538 <1> jnz short csftdf2_write_fat_file_secs_@
4539 <1> push ebx
4540 <1> call add_new_cluster
4541 <1> pop ebx
4542 <1> jc short csftdf2_write_fat_file_secs_5
4543 <1> mov [csftdf_df_cluster], eax
4544 <1> csftdf2_write_fat_file_secs_@:
4545 <1> ;;;
4546 <1> %endif
4547 <1>
4548 0000B991 A1[90820100] <1> mov eax, [csftdf_w_size]
4549 0000B996 29D2 <1> sub edx, edx
4550 <1> ;movzx ecx, word [esi+LD_BPB+BytesPerSec]
4551 <1> ; 19/12/2025
4552 0000B998 B900020000 <1> mov ecx, 512
4553 0000B99D 01C8 <1> add eax, ecx
4554 0000B99F 48 <1> dec eax
4555 0000B9A0 F7F1 <1> div ecx
4556 0000B9A2 89C1 <1> mov ecx, eax ; sector count
4557 0000B9A4 A1[88820100] <1> mov eax, [csftdf_df_cluster]
4558 <1>
4559 <1> ; EBX = memory block address (current)
4560 <1>
4561 0000B9A9 E8DF0E0000 <1> call write_fat_file_sectors
4562 0000B9AE 7254 <1> jc short csftdf2_write_fat_file_secs_4
4563 <1>
4564 <1> ; EBX = next memory address
4565 <1>
4566 0000B9B0 A1[98820100] <1> mov eax, [csftdf_df_wbytes]
4567 0000B9B5 0305[90820100] <1> add eax, [csftdf_w_size]
4568 0000B9BB 8B15[78820100] <1> mov edx, [csftdf_filesizes]
4569 0000B9C1 39D0 <1> cmp eax, edx
4570 0000B9C3 733F <1> jnb short csftdf2_write_fat_file_secs_4

```

```

4571 0000B9C5 A3[98820100] <1> mov [csftdf_df_wbytes], eax
4572 <1> ;
4573 0000B9CA A3[3A820100] <1> mov [DestinationFile_DirEntry+DirEntry_FileSize], eax
4574 <1>
4575 <1>
4576 0000B9CF 53 <1> push ebx ; *
4577 <1>
4578 <1> ; 30/08/2024
4579 0000B9D0 A1[88820100] <1> mov eax, [csftdf_df_cluster] ; last cluster
4580 <1>
4581 0000B9D5 803D[76820100]01 <1> cmp byte [DestinationFileFound], 1
4582 0000B9DC 720B <1> jb short csftdf2_write_fat_file_secs_2
4583 <1>
4584 <1> ; get next cluster (csftdf_w_size! bytes)
4585 <1> ;mov eax, [csftdf_df_cluster]
4586 0000B9DE E896050000 <1> call get_next_cluster
4587 0000B9E3 7317 <1> jnc short csftdf2_write_fat_file_secs_3
4588 <1>
4589 0000B9E5 21C0 <1> and eax, eax ; end of cluster chain!?
4590 0000B9E7 751C <1> jnz short csftdf2_write_fat_file_secs_5 ; disk error !
4591 <1>
4592 <1> csftdf2_write_fat_file_secs_2:
4593 <1> ;mov eax, [csftdf_df_cluster] ; last cluster
4594 <1> ; 30/08/2024
4595 0000B9E9 E80E0E0000 <1> call add_new_cluster
4596 0000B9EE 7215 <1> jc short csftdf2_write_fat_file_secs_5
4597 <1>
4598 <1> ; NOTE: Destination file size may be bigger than
4599 <1> ; source file size when the last reading fails after here.
4600 <1> ; (The last -empty- cluster of destination file must be
4601 <1> ; truncated and LMDT must be current date&time for partial
4602 <1> ; copy result!)
4603 0000B9F0 8B15[90820100] <1> mov edx, [csftdf_w_size] ; < = bytes per cluster
4604 0000B9F6 0115[3A820100] <1> add [DestinationFile_DirEntry+DirEntry_FileSize], edx
4605 <1>
4606 <1> csftdf2_write_fat_file_secs_3:
4607 <1> ; 30/08/2024
4608 <1> ;pop ebx ; *
4609 0000B9FC 29D2 <1> sub edx, edx ; 0
4610 0000B9FE A3[88820100] <1> mov [csftdf_df_cluster], eax ; next cluster
4611 <1>
4612 <1> ; 30/08/2024
4613 0000BA03 5B <1> pop ebx ; *
4614 <1>
4615 <1> csftdf2_write_fat_file_secs_4:
4616 0000BA04 C3 <1> retn
4617 <1>
4618 <1> csftdf2_write_fat_file_secs_5:
4619 0000BA05 5B <1> pop ebx ; *
4620 0000BA06 B812000000 <1> mov eax, 18 ; Write error !
4621 <1> ; 16/10/2016 (1Dh -> 18)
4622 0000BA0B C3 <1> retn
4623 <1>
4624 <1> csftdf2_save_file:
4625 <1> ; 19/12/2025
4626 <1> ; 31/08/2024
4627 <1> ; 26/08/2024, 30/08/2024
4628 <1> ; 09/12/2017
4629 <1> ; 19/03/2016, 25/03/2016
4630 <1> ; 18/03/2016
4631 0000BA0C 8B35[A4820100] <1> mov esi, [csftdf_df_drv_dt] ; logical dos drv desc. tbl.
4632 0000BA12 8B1D[7C820100] <1> mov ebx, [csftdf_sf_mem_addr] ; memory block address
4633 <1>
4634 <1> ; 19/12/2025
4635 <1> %if 0
4636 <1> cmp byte [esi+LD_FATType], 0
4637 <1> ;jna csftdf2_save_fs_file
4638 <1> ; 29/07/2022
4639 <1> ja short csftdf2_save_fat_file
4640 <1> jmp csftdf2_save_fs_file
4641 <1> %endif
4642 <1>
4643 <1> csftdf2_save_fat_file:
4644 <1>
4645 <1> ; 31/08/2024
4646 <1> %if 1
4647 <1> ; 30/08/2024
4648 0000BA18 53 <1> push ebx ; *
4649 <1> ;;;
4650 <1> ; 27/08/2024
4651 <1> ;call csftdf2_update_df_fclust
4652 0000BA19 E8F2010000 <1> call csftdf2_update_df_fclust_@ ; 31/08/2024
4653 0000BA1E 5B <1> pop ebx ; *
4654 0000BA1F 7236 <1> jc short csftdf2_save_fat_file_err
4655 <1> ; esi = [csftdf_df_drv_dt] ; (*)
4656 <1> ;;;
4657 <1> %endif
4658 <1>
4659 0000BA21 803D[9C820100]00 <1> cmp byte [csftdf_percentage], 0
4660 0000BA28 7726 <1> ja short csftdf2_save_fat_file_0 ; 30/08/2024
4661 <1>
4662 0000BA2A 53 <1> push ebx ; **
4663 <1>
4664 <1> ; Set cursor position
4665 <1> ; AH= 02h, BH= Page Number, DH= Row, DL= Column
4666 0000BA2B 8A3D[9D820100] <1> mov bh, [csftdf_videopage]
4667 0000BA31 668B15[9E820100] <1> mov dx, [csftdf_cursorpos]
4668 0000BA38 B402 <1> mov ah, 2
4669 0000BA3A E8E55CFFFF <1> call _int10h
4670 <1>
4671 0000BA3F BE[E2340100] <1> mov esi, msg_writing
4672 0000BA44 E861B3FFFF <1> call print_msg
4673 <1>
4674 <1> csftdf2_save_fat_file_next:
4675 0000BA49 5B <1> pop ebx ; **
4676 <1>
4677 <1> ; 31/08/2024
4678 0000BA4A 8B35[A4820100] <1> mov esi, [csftdf_df_drv_dt]
4679 <1>
4680 <1> csftdf2_save_fat_file_0: ; 30/08/2024
4681 <1>
4682 <1> csftdf2_save_fat_file_1:
4683 0000BA50 E822FFFFFF <1> call csftdf2_write_file_sectors ; 19/03/2016
4684 <1> ;jc csftdf2_rw_error ; eocc! or disk error!
4685 <1> ; 29/07/2022
4686 0000BA55 7305 <1> jnc short csftdf2_6
4687 <1> csftdf2_save_fat_file_err: ; 27/08/2024
4688 0000BA57 E979010000 <1> jmp csftdf2_rw_error
4689 <1>
4690 <1> csftdf2_6:
4691 0000BA5C 09D2 <1> or edx, edx ; edx > 0 -> EOF
4692 0000BA5E 756C <1> jnz short csftdf2_save_fat_file_3 ; 25/03/2016
4693 <1>
4694 0000BA60 803D[9C820100]00 <1> cmp byte [csftdf_percentage], 0

```

```

4695 0000BA67 76E7      <1>      jna      short csftdf2_save_fat_file_1
4696                                <1>
4697 0000BA69 B020      <1>      mov      al, 20h
4698 0000BA6B BF[EE340100] <1>      mov      edi, percentagestr
4699 0000BA70 AA        <1>      stosb
4700 0000BA71 AA        <1>      stosb
4701 0000BA72 A1[98820100] <1>      mov      eax, [csftdf_df_wbytes]
4702                                <1>      ;mov      edx, 100
4703                                <1>      ; 29/07/2022
4704 0000BA77 31D2      <1>      xor      edx, edx
4705 0000BA79 B264      <1>      mov      dl, 100
4706 0000BA7B F7E2      <1>      mul      edx
4707 0000BA7D 8B0D[78820100] <1>      mov      ecx, [csftdf_filesize]
4708 0000BA83 F7F1      <1>      div      ecx
4709 0000BA85 B10A      <1>      mov      cl, 10
4710 0000BA87 F6F1      <1>      div      cl
4711 0000BA89 80C430    <1>      add      ah, '0'
4712 0000BA8C 8827      <1>      mov      [edi], ah
4713 0000BA8E 20C0      <1>      and      al, al
4714 0000BA90 740A      <1>      jz      short csftdf2_save_fat_file_2
4715 0000BA92 4F        <1>      dec      edi
4716                                <1>      ;cbw
4717 0000BA93 30E4      <1>      xor      ah, ah ; 09/12/2017
4718 0000BA95 F6F1      <1>      div      cl
4719 0000BA97 80C430    <1>      add      ah, '0'
4720 0000BA9A 8827      <1>      mov      [edi], ah
4721                                <1>      ;and      al, al
4722                                <1>      ;jz      short csftdf2_save_fat_file_2
4723                                <1>      ;dec      edi
4724                                <1>      ;mov      [edi], '1' ; 100%
4725                                <1>
4726                                <1> csftdf2_save_fat_file_2:
4727 0000BA9C 53        <1>      push     ebx ; *
4728                                <1>
4729 0000BA9D E802000000 <1>      call     csftdf2_print_wr_percentage ; 25/03/2016
4730                                <1>
4731 0000BAA2 EBA5        <1>      jmp      csftdf2_save_fat_file_next
4732                                <1>
4733                                <1> csftdf2_print_wr_percentage:
4734                                <1>      ; Set cursor position
4735                                <1>      ; AH= 02h, BH= Page Number, DH= Row, DL= Column
4736 0000BAA4 8A3D[9D820100] <1>      mov      bh, [csftdf_videopage]
4737 0000BAAA 668B15[9E820100] <1>      mov      dx, [csftdf_cursorpos]
4738 0000BAB1 B402      <1>      mov      ah, 2
4739 0000BAB3 E86C5CFFFF <1>      call     _int10h
4740                                <1>
4741 0000BAB8 BE[E2340100] <1>      mov      esi, msg_writing
4742 0000BABD E8E8B2FFFF <1>      call     print_msg
4743                                <1>
4744 0000BAC2 BE[EE340100] <1>      mov      esi, percentagestr
4745                                <1>      ;call     print_msg
4746                                <1>      ;retn
4747 0000BAC7 E9DEB2FFFF <1>      jmp      print_msg
4748                                <1>
4749                                <1> csftdf2_save_fat_file_3:
4750 0000BACC 803D[9C820100]00 <1>      cmp      byte [csftdf_percentage], 0
4751                                <1>      ;jna      csftdf2_save_fat_file_4 ; 25/03/2016
4752                                <1>      ; 29/07/2022
4753                                <1>      ;ja      short csftdf2_7
4754                                <1>      ;jmp     csftdf2_save_fat_file_4
4755                                <1>      ; 31/08/2024
4756 0000BAD3 7611      <1>      jna      short csftdf2_save_fat_file_4
4757                                <1>
4758                                <1> csftdf2_7:
4759                                <1>      ; "100%"
4760 0000BAD5 BF[EE340100] <1>      mov      edi, percentagestr
4761 0000BADA B031      <1>      mov      al, '1'
4762 0000BADC AA        <1>      stosb
4763 0000BADD B030      <1>      mov      al, '0'
4764 0000BADF AA        <1>      stosb
4765 000BAE0 AA        <1>      stosb
4766                                <1>
4767 0000BAE1 E8BEFFFFF <1>      call     csftdf2_print_wr_percentage
4768                                <1>
4769                                <1> csftdf2_save_fat_file_4:
4770 0000BAE6 803D[76820100]00 <1>      cmp      byte [DestinationFileFound], 0
4771 0000BAED 7645      <1>      jna      short csftdf2_save_fat_file_6
4772                                <1>
4773 0000BAEF 8B35[A4820100] <1>      mov      esi, [csftdf_df_drv_dt] ; 31/03/2016
4774                                <1>
4775 0000BAF5 A1[88820100] <1>      mov      eax, [csftdf_df_cluster] ; last cluster
4776 0000BAFA E87A040000 <1>      call     get_next_cluster
4777 0000BAFF 7233      <1>      jc      short csftdf2_save_fat_file_6 ; eoccl or disk error!
4778                                <1>
4779 0000BB01 A1[88820100] <1>      mov      eax, [csftdf_df_cluster] ; last cluster
4780                                <1>      ; 29/08/2024
4781 0000BB06 31C9      <1>      xor      ecx, ecx
4782                                <1>      ;mov      [FAT_ClusterCounter], ecx ; 0 ; reset
4783 0000BB08 49        <1>      dec      ecx ; 0FFFFFFFh
4784                                <1>      ;shr      ecx, 4 ; 28 bit ; 0FFFFFFFh
4785                                <1>      ; 29/08/2024
4786                                <1>      ; (update_cluster will save 28 bit cluster value)
4787                                <1>      ;mov      ecx, 0FFFFFFFh
4788 0000BB09 E852070000 <1>      call     update_cluster
4789 0000BB0E 7224      <1>      jc      short csftdf2_save_fat_file_6 ; really last cluster!?
4790                                <1>
4791 0000BB10 A3[88820100] <1>      mov      [csftdf_df_cluster], eax ; next cluster
4792                                <1>
4793                                <1>      ; byte [FAT_BuffValidData] = 2
4794 0000BB15 E8BA090000 <1>      call     save_fat_buffer
4795 0000BB1A 730E      <1>      jnc      short csftdf2_save_fat_file_5
4796                                <1>
4797 0000BB1C 8B15[78820100] <1>      mov      edx, [csftdf_filesize]
4798 0000BB22 8915[3A820100] <1>      mov      [DestinationFile_DirEntry+DirEntry_FileSize], edx
4799 0000BB28 EB57      <1>      jmp      short csftdf2_save_fat_file_err3
4800                                <1>
4801                                <1> csftdf2_save_fat_file_5:
4802 0000BB2A A1[88820100] <1>      mov      eax, [csftdf_df_cluster]
4803                                <1>
4804                                <1>      ; EAX = First cluster to be truncated/unlinked
4805                                <1>      ; ESI = Logical dos drive description table address
4806 0000BB2F E8DA0B0000 <1>      call     truncate_cluster_chain
4807                                <1>
4808                                <1> csftdf2_save_fat_file_6:
4809                                <1>      ; 28/03/2016
4810 0000BB34 BE[A9810100] <1>      mov      esi, SourceFile_DirEntry+DirEntry_Attr ; +11 to + 18
4811 0000BB39 BF[29820100] <1>      mov      edi, DestinationFile_DirEntry+DirEntry_Attr ; +11 to + 18
4812 0000BB3E A4        <1>      movsb ; +11
4813 0000BB3F A5        <1>      movsd ; +12 .. +15
4814 0000BB40 66A5      <1>      movsw ; +16 .. +17
4815                                <1>      ; + 18
4816 0000BB42 83C604    <1>      add      esi, 4
4817 0000BB45 83C704    <1>      add      edi, 4
4818 0000BB48 A5        <1>      movsd ; DirEntry_WrtTime ; +22 .. +25

```

```

4819                                     <1>
4820 0000BB49 8B15[78820100]          <1>      mov     edx, [csftdf_filesize]
4821 0000BB4F 8915[3A820100]          <1>      mov     [DestinationFile_DirEntry+DirEntry_FileSize], edx
4822                                     <1>
4823 0000BB55 E892F1FFFF              <1>      call    convert_current_date_time
4824                                     <1>      ; DX = Date in dos dir entry format
4825                                     <1>      ; AX = Time in dos dir entry format
4826 0000BB5A EB4C                    <1>      jmp     short csftdf2_save_fat_file_7
4827                                     <1>
4828                                     <1>      ; 31/08/2024
4829                                     <1> ;csftdf2_save_fat_file_err1:
4830                                     <1> ;pop     ebx ; *
4831                                     <1>
4832                                     <1> csftdf2_save_fat_file_err2:
4833 0000BB5C A1[98820100]            <1>      mov     eax, [csftdf_df_wbytes]
4834 0000BB61 8B15[3A820100]          <1>      mov     edx, [DestinationFile_DirEntry+DirEntry_FileSize]
4835 0000BB67 39C2                    <1>      cmp     edx, eax
4836 0000BB69 7616                    <1>      jna     short csftdf2_save_fat_file_err3
4837 0000BB6B A1[88820100]            <1>      mov     eax, [csftdf_df_cluster] ; last (empty) cluster
4838                                     <1>      ; ESI = Logical dos drive description table address
4839 0000BB70 E8990B0000              <1>      call    truncate_cluster_chain
4840 0000BB75 720A                    <1>      jc      short csftdf2_save_fat_file_err3
4841 0000BB77 A1[98820100]            <1>      mov     eax, [csftdf_df_wbytes]
4842 0000BB7C A3[3A820100]            <1>      mov     [DestinationFile_DirEntry+DirEntry_FileSize], eax
4843                                     <1> csftdf2_save_fat_file_err3:
4844 0000BB81 E866F1FFFF              <1>      call    convert_current_date_time
4845                                     <1>      ; DX = Date in dos dir entry format
4846                                     <1>      ; AX = Time in dos dir entry format
4847 0000BB86 C605[2B820100]00        <1>      mov     byte [DestinationFile_DirEntry+DirEntry_CrtTimeTenth], 0
4848 0000BB8D 66A3[2C820100]          <1>      mov     [DestinationFile_DirEntry+DirEntry_CrtTime], ax
4849 0000BB93 668915[2E820100]        <1>      mov     [DestinationFile_DirEntry+DirEntry_CrtDate], dx
4850 0000BB9A 66A3[34820100]          <1>      mov     [DestinationFile_DirEntry+DirEntry_WrtTime], ax
4851 0000BBA0 668915[36820100]        <1>      mov     [DestinationFile_DirEntry+DirEntry_WrtDate], dx
4852 0000BBA7 F9                      <1>      stc
4853                                     <1> csftdf2_save_fat_file_7:
4854 0000BBA8 9C                      <1>      pushf
4855 0000BBA9 668915[30820100]          <1>      mov     [DestinationFile_DirEntry+DirEntry_LastAccDate], dx
4856 0000BBB0 BE[1E820100]            <1>      mov     esi, DestinationFile_DirEntry
4857 0000BBB5 BF00000800              <1>      mov     edi, Directory_Buffer
4858 0000BBBA 0FB70D[46820100]          <1>      movzx   ecx, word [DestinationFile_DirEntryNumber] ; (<2048)
4859                                     <1>      ;shl     cx, 5 ; 32 * directory entry number
4860                                     <1>      ; 29/07/2022
4861 0000BBC1 C1E105                  <1>      shl     ecx, 5
4862 0000BBC4 01CF                    <1>      add     edi, ecx
4863                                     <1>      ;mov     ecx, 8
4864                                     <1>      ;mov     cx, 8
4865                                     <1>      ; 31/08/2024
4866                                     <1>      ; 29/07/2022
4867 0000BBC6 29C9                    <1>      sub     ecx, ecx
4868                                     <1>      ;sub     ch, ch
4869 0000BBC8 B108                    <1>      mov     cl, 8
4870 0000BBCA F3A5                    <1>      rep     movsd
4871 0000BBCC 9D                      <1>      popf
4872 0000BBCE 730B                    <1>      jnc     short csftdf2_write_file_OK
4873                                     <1>
4874                                     <1> csftdf2_write_error:
4875                                     <1>      ; 18/03/2016
4876                                     <1>      ;mov     al, 1Dh ; write error
4877                                     <1>      ; 31/08/2024
4878 0000BBCE B012                    <1>      mov     al, ERR_DRV_WRITE ; 18 ; write error
4879 0000BBD1 EB02                    <1>      jmp     short csftdf2_rw_error
4880                                     <1>
4881                                     <1>      ; 16/03/2016
4882                                     <1> csftdf2_read_error:
4883 0000BBD3 B011                    <1>      mov     al, 17 ; ; Drive not ready or read error!
4884                                     <1> csftdf2_rw_error:
4885 0000BBD5 A2[75820100]            <1>      mov     [csftdf_rw_err], al
4886                                     <1>
4887                                     <1> csftdf2_write_file_OK:
4888                                     <1>      ; 18/03/2016
4889 0000BBDA C605[897E0100]02        <1>      mov     byte [DirBuff_ValidData], 2
4890 0000BBE1 E89BF1FFFF              <1>      call    save_directory_buffer
4891                                     <1>
4892                                     <1>      ; Update last modification date&time of destination
4893                                     <1>      ; file's (parent) directory
4894 0000BBE6 E81FF2FFFF              <1>      call    update_parent_dir_lmdt
4895                                     <1>      ;
4896 0000BBEB A1[7C820100]            <1>      mov     eax, [csftdf_sf_mem_addr] ; start address
4897                                     <1>
4898 0000BBF0 21C0                    <1>      and     eax, eax
4899 0000BBF2 7509                    <1>      jnz     short csftdf2_dealloc_mblock
4900                                     <1>
4901                                     <1>      ; 31/08/2024
4902                                     <1>      ;mov     ch, al ; 0 (cluster r/w, not full loading)
4903                                     <1> csftdf2_dealloc_retn:
4904                                     <1>      ;mov     cl, [csftdf_rw_err]
4905                                     <1>      ;mov     eax, [csftdf_df_cluster]
4906                                     <1>      ; 31/08/2024
4907 0000BBF4 A0[75820100]            <1>      mov     al, [csftdf_rw_err]
4908 0000BBF9 3C01                    <1>      cmp     al, 1
4909 0000BBFB F5                      <1>      cmc
4910                                     <1>      ; if al > 0 -> cf = 1
4911                                     <1> csftdf2_write_df_error:      ; 31/08/2024
4912                                     <1> csftdf2_write_df_clust_@@: ; 31/08/2024
4913 0000BBFC C3                      <1>      retn
4914                                     <1>
4915                                     <1> csftdf2_dealloc_mblock:
4916 0000BBFD 8B0D[80820100]          <1>      mov     ecx, [csftdf_sf_mem_bsize] ; block size
4917 0000BC03 E84BA3FFFF              <1>      call    deallocate_memory_block
4918                                     <1>      ; 31/08/2024
4919                                     <1>      ;mov     ch, 0FFh ; (File was full loaded at memory)
4920 0000BC08 EBFA                    <1>      jmp     short csftdf2_dealloc_retn
4921                                     <1>
4922                                     <1> ;;;;
4923                                     <1> csftdf2_update_df_fclust:
4924                                     <1>      ; 31/08/2024
4925                                     <1>      ; 27/08/2024 - TRDOS 386 v2.0.9
4926                                     <1>      ; update (directory entry of)
4927                                     <1>      ; the new (created) destination file
4928                                     <1>      ; with new first cluster (it was 0 before)
4929                                     <1>
4930 0000BC0A 8B35[A4820100]          <1>      mov     esi, [csftdf_df_drv_dt] ; (!)
4931                                     <1>
4932                                     <1> csftdf2_update_df_fclust_@: ; 31/08/2024
4933 0000BC10 A1[88820100]            <1>      mov     eax, [csftdf_df_cluster]
4934 0000BC15 21C0                    <1>      and     eax, eax
4935 0000BC17 75E3                    <1>      jnz     short csftdf2_write_df_clust_@@
4936                                     <1>      ; if eax = 0, add new cluster
4937                                     <1>      ; 31/08/2024
4938                                     <1> %if 0
4939                                     <1>      call    get_first_free_cluster
4940                                     <1>      jc      short csftdf2_write_df_error
4941                                     <1>      ; EAX >= 2 and EAX < FFFFFFFFh is valid
4942                                     <1>

```

```

4943      <1>      cmp     eax, 0FFFFFFFh ; no free space
4944      <1>      jne     short csftdf2_write_df_clust_@
4945      <1>      inc     eax ; 0
4946      <1>      stc
4947      <1>      mov     al, 27h ; insufficient disk space
4948      <1>      csftdf2_write_df_error:
4949      <1>      retn
4950      <1>      %else
4951      <1>      ; 31/08/2024
4952      <1>      call    add_new_cluster
4953      <1>      jc      short csftdf2_write_df_error
4954      <1>      %endif
4955      <1>
4956      <1>      csftdf2_write_df_clust_@:
4957      <1>      ; 31/08/2024
4958      <1>      mov     [csftdf_df_cluster], eax
4959      <1>      ;
4960      <1>      ; eax = (new) first cluster
4961      <1>      push    esi
4962      <1>      mov     esi, DestinationFile_DirEntry
4963      <1>      movzx   edi, word [DestinationFile_DirEntryNumber]
4964      <1>      shl     edi, 5 ; 32 * index number
4965      <1>      add     edi, Directory_Buffer
4966      <1>      rol     eax, 16
4967      <1>      mov     [esi+DirEntry_FstClusHI], ax
4968      <1>      ror     eax, 16
4969      <1>      mov     [esi+DirEntry_FstClusLO], ax
4970      <1>      mov     ecx, 8 ; 32 bytes
4971      <1>      rep     movsd
4972      <1>      pop     esi
4973      <1>      ;
4974      <1>      mov     byte [DirBuff_ValidData], 2 ; change sign
4975      <1>      ;
4976      <1>      ; 31/08/2024
4977      <1>      %if 0
4978      <1>      call    save_directory_buffer
4979      <1>      jc      short csftdf2_write_df_error
4980      <1>      ;
4981      <1>      ; +1 cluster is in use (as first cluster)
4982      <1>      mov     ebx, [esi+LD_FreeSectors]
4983      <1>      inc     ebx ; 0FFFFFFFh ? invalid ?
4984      <1>      jz      short csftdf2_write_df_clust_@@ ; yes
4985      <1>      xor     ebx, ebx
4986      <1>      mov     bl, [esi+LD_BPB+BPB_SecPerClust]
4987      <1>      sub     [esi+LD_FreeSectors], ebx
4988      <1>      ;
4989      <1>      csftdf2_write_df_clust_@@:
4990      <1>      retn
4991      <1>      ;;;
4992      <1>      %else
4993      <1>      ; 31/08/2024
4994      <1>      jmp     save_directory_buffer
4995      <1>      %endif
4996      <1>
4997      <1>      csftdf2_save_fs_file:
4998      <1>      ; 16/10/2016 (1dh -> 18)
4999      <1>      ; temporary - (21/03/2016)
5000      <1>      mov     eax, 18 ; write error
5001      <1>      ; 29/07/2022
5002      <1>      xor     eax, eax
5003      <1>      mov     al, 18
5004      <1>      stc
5005      <1>      retn
5006      <1>
5007      <1>      create_file:
5008      <1>      ; 19/12/2025 (TRDOS 386 v2.0.10)
5009      <1>      ; 31/08/2024
5010      <1>      ; 29/08/2024
5011      <1>      ; 27/08/2024
5012      <1>      ; 26/08/2024 (TRDOS 386 v2.0.9)
5013      <1>      ; 30/07/2022
5014      <1>      ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
5015      <1>      ; 16/10/2016
5016      <1>      ; 24/03/2016, 31/03/2016
5017      <1>      ; 20/03/2016, 21/03/2016, 23/03/2016
5018      <1>      ; 19/03/2016 (TRDOS 396 = TRDOS v2.0)
5019      <1>      ; 03/09/2011 (FILE.ASM, 'proc_create_file')
5020      <1>      ; 09/08/2010
5021      <1>      ;
5022      <1>      ; INPUT ->
5023      <1>      ; EAX = File Size
5024      <1>      ; ESI = ASCIIZ File Name
5025      <1>      ; CL = File Attributes
5026      <1>      ; EBX = FFFFFFFFh -> create empty file
5027      <1>      ; (only for FAT fs)
5028      <1>      ; OUTPUT ->
5029      <1>      ; EAX = New file's first cluster
5030      <1>      ; (0 for empty file) ; 29/08/2024
5031      <1>      ; ESI = Logical Dos Drv Descr. Table Addr.
5032      <1>      ; EBX = CreateFile_Size address
5033      <1>      ; ECX = Sectors per cluster (<256)
5034      <1>      ; EDX = Directory Entry Index/Number (<65536)
5035      <1>      ; ; 29/08/2024
5036      <1>      ; EBX = File Size (0 for a new, empty file)
5037      <1>      ; ECX = Directory Entry Index/Number (<2048)
5038      <1>      ; (in directory cluster, not in directory)
5039      <1>      ; EDX = Directory Cluster Number (of the file)
5040      <1>      ;
5041      <1>      ; cf = 1 -> error code in AL (EAX)
5042      <1>      ;
5043      <1>      ; (Modified registers: eax, ebx, ecx, edx, esi, edi)
5044      <1>      ;
5045      <1>      ;
5046      <1>      ; test    cl, 18h (directory or volume name)
5047      <1>      ; jnz     short loc_createfile_access_denied
5048      <1>      and     cl, 07h ; S, H, R
5049      <1>      mov     [createfile_attr], cl
5050      <1>
5051      <1>      mov     ecx, ebx
5052      <1>      mov     ebx, esi ; ASCIIZ File Name address
5053      <1>      sub     edx, edx
5054      <1>      mov     dh, [Current_Drv]
5055      <1>      mov     esi, Logical_DOSDisks
5056      <1>      add     esi, edx
5057      <1>
5058      <1>      mov     [createfile_UpdatePDir], dl ; 0 ; 31/03/2016
5059      <1>
5060      <1>      ; LD_DiskType = 0 for write protection (read only)
5061      <1>      cmp     byte [esi+LD_DiskType], 1 ; 0 = Invalid
5062      <1>      jnb     short loc_createfile_check_file_sytem
5063      <1>      ; 16/10/2016 (TRDOS Error code: 30, disk write protected)
5064      <1>      mov     eax, 30 ; 13h, MSDOS err : Disk write-protected
5065      <1>      mov     dx, 0
5066      <1>      ; 29/07/2022

```



```

5067 0000BC8B 31D2      <1>      xor     edx, edx ; 0
5068 0000BC8D 31C0      <1>      xor     eax, eax ; 0
5069 0000BC8F B01E      <1>      mov     al, 30
5070 0000BC91 F9          <1>      stc
5071          <1>      ; err retn: EDX = 0, EBX = File name offset
5072          <1>      ; ESI -> Dos drive description table address
5073 0000BC92 C3          <1>      retn
5074          <1>
5075          <1>      ;loc_createfile_access_denied:
5076          <1>      ; mov     eax, 05h ; access denied (invalid attributes input)
5077          <1>      ; stc
5078          <1>      ; retn
5079          <1>
5080          <1>      loc_createfile_check_file_sytem:
5081          <1>      ; 27/08/2024
5082 0000BC93 A3[B0820100] <1>      mov     [createfile_size], eax
5083          <1>
5084          <1>      ; 19/12/2025
5085          <1>      %if 0
5086          <1>      cmp     byte [esi+LD_FATType], 1
5087          <1>      jnb     short loc_createfile_chk_empty_FAT_file_sign1
5088          <1>
5089          <1>      ; 27/08/2024
5090          <1>      ;mov     [createfile_size], eax
5091          <1>      ; ESI = Logical Dos Drive Description Table address
5092          <1>      ; EBX = ASCIIIZ File Name address
5093          <1>      jmp     create_fs_file
5094          <1>      %endif
5095          <1>
5096          <1>      loc_createfile_chk_empty_FAT_file_sign1:
5097          <1>      ; ECX = FFFFFFFFh -> create empty file if drive has FAT fs
5098 0000BC98 41          <1>      inc     ecx
5099 0000BC99 7506          <1>      jnz     short loc_createfile_chk_empty_FAT_file_sign2
5100          <1>
5101 0000BC9B 890D[B0820100] <1>      mov     [createfile_size], ecx ; 0 ; empty file
5102          <1>
5103          <1>      loc_createfile_chk_empty_FAT_file_sign2:
5104          <1>      ; 19/12/2025
5105          <1>      ; 23/03/2016
5106          <1>      ;mov     cx, [esi+LD_BPB+BytesPerSec]
5107          <1>      ;mov     [createfile_BytesPerSec], cx
5108          <1>
5109          <1>      ; EBX = ASCIIIZ File Name address
5110 0000BCA1 0FB65613      <1>      movzx    edx, byte [esi+LD_BPB+SecPerClust]
5111 0000BCA5 8815[C5820100] <1>      mov     [createfile_SecPerClust], dl
5112 0000BCAB 8B4E74      <1>      mov     ecx, [esi+LD_FreeSectors]
5113 0000BCAE 39D1          <1>      cmp     ecx, edx ; byte [createfile_SecPerClust]
5114 0000BCB0 7306          <1>      jnb     short loc_create_fat_file
5115          <1>
5116          <1>      loc_createfile_insufficient_disk_space:
5117 0000BCB2 B827000000      <1>      mov     eax, 27h
5118          <1>      loc_createfile_gffc_retn:
5119 0000BCB7 C3          <1>      retn
5120          <1>
5121          <1>      loc_create_fat_file:
5122          <1>      mov     [createfile_Name_Offset], ebx
5123 0000BCBE 890D[AC820100] <1>      mov     [createfile_FreeSectors], ecx
5124          <1>
5125          <1>      ; 27/08/2024
5126          <1>      ;loc_createfile_gffc_1:
5127          <1>      ;call    get_first_free_cluster
5128          <1>      ;jc     short loc_createfile_gffc_retn
5129          <1>      ;
5130          <1>      ;mov     [createfile_FFCluster], eax
5131          <1>
5132          <1>      loc_createfile_locate_ffc_on_directory:
5133          <1>      ; Current directory fcluster <=> Directory buffer cluster
5134          <1>      ; Current directory will be reloaded by
5135          <1>      ; 'locate_current_dir_file' procedure
5136          <1>      ;
5137          <1>      ; ESI = Logical Dos Drv Desc. Table Address
5138 0000BCC4 56          <1>      push    esi ; *
5139 0000BCC5 31C0      <1>      xor     eax, eax
5140          <1>
5141 0000BCC7 A3[7E7E0100] <1>      mov     [FAT_ClusterCounter], eax ; 0
5142          <1>      ; 21/03/2016
5143 0000BCCC A2[CC820100] <1>      mov     [createfile_wfc], al ; 0
5144          <1>
5145 0000BCD1 89C1          <1>      mov     ecx, eax
5146 0000BCD3 6649          <1>      dec     cx ; FFFFh
5147          <1>      ; CX = FFFFh -> find first deleted or free entry
5148          <1>      ; ESI would be ASCIIIZ filename address if the call
5149          <1>      ; would not be for first free or deleted dir entry
5150 0000BCD5 E81EE9FFFF      <1>      call    locate_current_dir_file
5151          <1>      ;jnc     loc_createfile_set_ff_dir_entry
5152          <1>      ; 29/07/2022
5153 0000BCDA 7205          <1>      jc     short loc_createfile_locate_file_err
5154          <1>      ; 26/08/2024
5155          <1>      ; cl = 0FFh, ch=0 or ch=0E5h
5156          <1>      ; 27/08/2024
5157          <1>      ;xor     ecx, ecx ; 26/08/2024
5158 0000BCDC E9DA000000      <1>      jmp     loc_createfile_set_ff_dir_entry
5159          <1>
5160          <1>      loc_createfile_locate_file_err: ; 29/07/2022
5161 0000BCE1 5E          <1>      pop     esi ; *
5162          <1>      ; ESI = Logical DOS Drv. Description Table Address
5163 0000BCE2 83F802      <1>      cmp     eax, 2
5164 0000BCE5 7402          <1>      je     short loc_createfile_add_new_cluster
5165          <1>      loc_createfile_locate_file_stc_retn:
5166 0000BCE7 F9          <1>      stc
5167 0000BCE8 C3          <1>      retn
5168          <1>
5169          <1>      loc_createfile_add_new_cluster:
5170 0000BCE9 803D[61770100]02 <1>      cmp     byte [Current_FATType], 2
5171          <1>      ;cmp     byte [esi+LD_FATType], 2
5172 0000BCF0 770C          <1>      ja     short loc_createfile_add_new_cluster_check_fsc
5173 0000BCF2 803D[60770100]01 <1>      cmp     byte [Current_Dir_Level], 1
5174          <1>      ;cmp     byte [esi+LD_CDirLevel], 1
5175 0000BCF9 7303          <1>      jnb     short loc_createfile_add_new_cluster_check_fsc
5176          <1>
5177          <1>      ;mov     eax, 12
5178 0000BCFB B00C          <1>      mov     al, 12 ; No more files
5179          <1>
5180          <1>      loc_createfile_anc_retn:
5181 0000BCFD C3          <1>      retn
5182          <1>
5183          <1>      loc_createfile_add_new_cluster_check_fsc:
5184 0000BCFE 8B0D[AC820100] <1>      mov     ecx, [createfile_FreeSectors]
5185 0000BD04 0FB605[C5820100] <1>      movzx    eax, byte [createfile_SecPerClust]
5186          <1>      ;shl     ax, 1 ; AX = 2 * AX
5187          <1>      ; 29/07/2022
5188 0000BD0B D1E0          <1>      shl     eax, 1
5189 0000BD0D 39C1          <1>      cmp     ecx, eax
5190 0000BD0F 72A1          <1>      jb     short loc_createfile_insufficient_disk_space

```

```

5191 <1>
5192 <1> loc_createfile_add_new_subdir_cluster:
5193 0000BD11 8B15[8E7E0100] <1> mov     edx, [DirBuff_Cluster]
5194 0000BD17 8915[B8820100] <1> mov     [createfile_LastDirCluster], edx
5195 <1>
5196 <1> ; 27/08/2024
5197 <1> ;;;
5198 0000BD1D E869040000 <1> call    get_first_free_cluster
5199 0000BD22 72D9 <1> jc      short loc_createfile_anc_retn
5200 0000BD24 A3[B4820100] <1> mov     [createfile_FFCluster], eax
5201 <1> ;;;
5202 <1> ; 27/08/2024
5203 <1> ;;;
5204 <1> ;mov     eax, [createfile_FFCluster]
5205 <1> ; eax = cluster address of the new directory sector
5206 <1> ;;;
5207 <1>
5208 0000BD29 E807040000 <1> call    load_FAT_sub_directory
5209 0000BD2E 72CD <1> jc      short loc_createfile_anc_retn
5210 <1>
5211 <1> ; 27/08/2024
5212 <1> ; [DirBuff_Cluster] = [createfile_FFCluster]
5213 <1>
5214 <1> pass_createfile_add_new_subdir_cluster:
5215 <1> ; clear directory buffer (new dir sector) ; 27/08/2024
5216 <1> ;movzx eax, word [esi+LD_BPB+BytesPerSec]
5217 <1> ;movzx eax, word [createfile_BytesPerSec] ; 23/03/2016
5218 <1> ; 19/12/2025
5219 <1> ;mov     eax, 512
5220 <1> ; ecx = directory buffer sector count
5221 <1> ; (sectors per cluster)
5222 <1> ;mul     ecx
5223 <1> ;mov     ecx, eax
5224 <1> ; (bytes per cluster or directory buffer size)
5225 <1> ;shr     ecx, 2 ; dword count
5226 <1> ; 19/12/2025
5227 0000BD30 C1E107 <1> shl     ecx, 7 ; 512/4 (dword count)
5228 <1>
5229 0000BD33 29C0 <1> sub     eax, eax ; 0
5230 0000BD35 F3AB <1> rep     stosd
5231 <1> ;
5232 0000BD37 C605[897E0100]02 <1> mov     byte [DirBuff_ValidData], 2
5233 0000BD3E E83EF0FFFF <1> call    save_directory_buffer
5234 0000BD43 72B8 <1> jc      short loc_createfile_anc_retn
5235 <1>
5236 <1> loc_createfile_save_added_subdir_cluster:
5237 0000BD45 A1[B8820100] <1> mov     eax, [createfile_LastDirCluster]
5238 0000BD4A 8B0D[B4820100] <1> mov     ecx, [createfile_FFCluster]
5239 0000BD50 E80B050000 <1> call    update_cluster
5240 0000BD55 7304 <1> jnc     short loc_createfile_save_fat_buffer_0
5241 0000BD57 09C0 <1> or      eax, eax ; EAX = 0 -> cluster value is 0 or eocc
5242 0000BD59 7518 <1> jnz     short loc_createfile_save_fat_buffer_stc_retn
5243 <1>
5244 <1> loc_createfile_save_fat_buffer_0:
5245 0000BD5B A1[B4820100] <1> mov     eax, [createfile_FFCluster]
5246 0000BD60 A3[B8820100] <1> mov     [createfile_LastDirCluster], eax
5247 <1> ;mov     ecx, 0FFFFFFFh ; 28 bit
5248 <1> ; 29/08/2024
5249 <1> ; (update_cluster will save 28 bit cluster value)
5250 <1> ;mov     ecx, 0FFFFFFFh
5251 0000BD65 29C9 <1> sub     ecx, ecx
5252 0000BD67 49 <1> dec     ecx ; 0FFFFFFFh
5253 0000BD68 E8F3040000 <1> call    update_cluster
5254 0000BD6D 7306 <1> jnc     short loc_createfile_save_fat_buffer_1
5255 0000BD6F 09C0 <1> or      eax, eax ; Was it EOF ?
5256 0000BD71 7402 <1> jz      short loc_createfile_save_fat_buffer_1 ; yes
5257 <1>
5258 <1> loc_createfile_save_fat_buffer_stc_retn:
5259 0000BD73 F9 <1> stc
5260 <1> loc_createfile_save_fat_buffer_retn:
5261 <1> loc_createfile_gffc_2_stc_retn:
5262 0000BD74 C3 <1> retn
5263 <1>
5264 <1> loc_createfile_save_fat_buffer_1:
5265 <1> ; byte [FAT_BuffValidData] = 2
5266 0000BD75 E85A070000 <1> call    save_fat_buffer
5267 0000BD7A 72F8 <1> jc      short loc_createfile_save_fat_buffer_retn
5268 <1>
5269 0000BD7C 803D[7E7E0100]01 <1> cmp     byte [FAT_ClusterCounter], 1
5270 0000BD83 7222 <1> jb      short loc_createfile_save_fat_buffer_2
5271 <1>
5272 <1> ; ESI = Logical DOS Drive Description Table address
5273 0000BD85 A1[7E7E0100] <1> mov     eax, [FAT_ClusterCounter]
5274 <1>
5275 0000BD8A C605[7E7E0100]00 <1> mov     byte [FAT_ClusterCounter], 0 ; 21/03/2016
5276 <1>
5277 0000BD91 66BB01FF <1> mov     bx, 0FF01h ; add free clusters
5278 0000BD95 E8C7070000 <1> call    calculate_fat_freespace
5279 <1>
5280 <1> ;inc     eax ; 0FFFFFFFh -> 0 ; recalculation is needed!
5281 <1> ;jnz     short loc_createfile_save_fat_buffer_2
5282 <1>
5283 <1> ; ecx > 0 -> Recalculation is needed
5284 0000BD9A 09C9 <1> or      ecx, ecx
5285 0000BD9C 7409 <1> jz      short loc_createfile_save_fat_buffer_2
5286 <1>
5287 0000BD9E 66BB00FF <1> mov     bx, 0FF00h ; recalculate free space
5288 0000BDA2 E8BA070000 <1> call    calculate_fat_freespace
5289 <1>
5290 <1> loc_createfile_save_fat_buffer_2:
5291 <1> ;call    update_parent_dir_lmdt
5292 <1>
5293 <1> ; 27/08/2024
5294 <1> ;loc_createfile_gffc_2:
5295 <1> ;call    get_first_free_cluster
5296 <1> ;jc      short loc_createfile_gffc_2_stc_retn
5297 <1> ;
5298 <1> ;mov     [createfile_FFCluster], eax
5299 <1> ;
5300 <1>
5301 0000BDA7 A1[B8820100] <1> mov     eax, [createfile_LastDirCluster]
5302 <1>
5303 0000BDAC E884030000 <1> call    load_FAT_sub_directory
5304 0000BDB1 72C1 <1> jc      short loc_createfile_gffc_2_stc_retn
5305 <1>
5306 0000BDB3 BF00000800 <1> mov     edi, Directory_Buffer
5307 <1>
5308 <1> ; 27/08/2024
5309 <1> ;sub     bx, bx ; directory entry index/number = 0
5310 0000BDB8 29DB <1> sub     ebx, ebx
5311 <1> ; 29/07/2022
5312 <1> ;sub     ecx, ecx
5313 <1>
5314 0000BDBA 56 <1> push    esi ; * ; 23/03/2016

```

```

5315 <1>
5316 <1> loc_createfile_set_ff_dir_entry:
5317 <1> ;;;
5318 <1> ; 29/08/2024
5319 0000BDBB A1[8E7E0100] <1> mov     eax, [DirBuff_Cluster]
5320 0000BDC0 A3[B8820100] <1> mov     [createfile_LastDirCluster], eax
5321 <1> ;;;
5322 <1>
5323 <1> ; 27/08/2024
5324 0000BDC5 66891D[C6820100] <1> mov     [createfile_DirIndex], bx
5325 <1> ; 29/07/2022
5326 <1> ;mov     [createfile_DirIndex], cx ; 0
5327 <1>
5328 <1> ;;;
5329 0000BDCC A1[B0820100] <1> mov     eax, [createfile_size]
5330 0000BDD1 09C0 <1> or      eax, eax
5331 0000BDD3 740A <1> jz      short loc_createfile_sffc
5332 <1> ; 31/08/2024
5333 0000BDD5 8B3424 <1> mov     esi, [esp] ; * ; LDDDT address
5334 0000BDD8 E8AE030000 <1> call    get_first_free_cluster
5335 0000BDDD 7295 <1> jc      short loc_createfile_gffc_2_stc_retn
5336 <1> loc_createfile_sffc:
5337 0000BDDF A3[B4820100] <1> mov     [createfile_FFCluster], eax
5338 0000BDE4 29C9 <1> sub     ecx, ecx ; 0
5339 <1> ;;;
5340 <1>
5341 <1> ; EDI = Directory entry address
5342 0000BDE6 8B35[A8820100] <1> mov     esi, [createfile_Name_Offset]
5343 <1> ;;;
5344 <1> ; 27/08/2024
5345 <1> ;mov     eax, [createfile_FFCluster]
5346 <1> ;;;
5347 0000BDEC A3[BC820100] <1> mov     [createfile_Cluster], eax ; 24/03/2016
5348 <1> ;mov     ch, 0FFh
5349 <1> ; 29/07/2022
5350 <1> ; ecx = 0
5351 0000BDF1 FECD <1> dec     ch ; 0 -> 0FFh
5352 0000BDF3 8A0D[C4820100] <1> mov     cl, [createfile_attrib] ; file attributes
5353 <1> ; CH > 0 -> File size is in [EBX]
5354 0000BDF9 BB[B0820100] <1> mov     ebx, createfile_size
5355 <1>
5356 0000BDFF E8ABEEFFFF <1> call    make_directory_entry
5357 <1>
5358 0000BE03 5E <1> pop     esi ; * ; ESI = Logical Dos Drv Desc. Table address
5359 <1>
5360 0000BE04 C605[897E0100]02 <1> mov     byte [DirBuff_ValidData], 2
5361 0000BE0B E871EFFFFF <1> call    save_directory_buffer
5362 0000BE10 7213 <1> jc      short loc_createfile_set_ff_dir_entry_retn
5363 <1>
5364 0000BE12 C605[CD820100]01 <1> mov     byte [createfile_UpdatePDir], 1 ; 31/03/2016
5365 <1>
5366 <1> loc_createfile_get_set_write_file_cluster:
5367 0000BE19 A1[B0820100] <1> mov     eax, [createfile_size]
5368 0000BE1E 09C0 <1> or      eax, eax ; 0 ?
5369 0000BE20 755B <1> jnz     short loc_createfile_get_set_wfc_cont ; no
5370 0000BE22 40 <1> inc     eax ; eax = 1
5371 <1>
5372 <1> ; 27/08/2024 (empty file)
5373 <1> ;;;
5374 <1> ; 23/03/2016
5375 <1> ;movzx   ebx, byte [createfile_SecPerClust]
5376 <1> ;movzx   ecx, word [esi+LD_BPB+BytesPerSec] ; 512
5377 <1> ;movzx   ecx, word [createfile_BytesPerSec] ; 512
5378 <1> ;;;
5379 0000BE23 EB72 <1> jmp     loc_createfile_set_cluster_count
5380 <1>
5381 <1> loc_createfile_set_ff_dir_entry_retn:
5382 0000BE25 C3 <1> retn
5383 <1>
5384 <1> loc_createfile_write_fcluster_to_disk:
5385 0000BE26 034668 <1> add     eax, [esi+LD_DATABegin] ; convert to physical address
5386 0000BE29 BB00000700 <1> mov     ebx, Cluster_Buffer
5387 <1> ; ESI = Logical DOS Drv. Desc. Tbl. address
5388 <1> ; EAX = Disk address
5389 <1> ; EBX = Sector Buffer
5390 <1> ; ECX = sectors per cluster
5391 0000BE2E E8A15B0000 <1> call    disk_write
5392 0000BE33 7211 <1> jc      short loc_createfile_dsk_wr_err
5393 <1>
5394 <1> loc_createfile_update_fat_cluster:
5395 <1> ; 21/03/2016
5396 0000BE35 803D[CC820100]00 <1> cmp     byte [createfile_wfc], 0
5397 0000BE3C 7711 <1> ja      short loc_createfile_update_fat_cluster_n1
5398 <1>
5399 0000BE3E FE05[CC820100] <1> inc     byte [createfile_wfc] ; 1
5400 0000BE44 EB1F <1> jmp     short loc_createfile_update_fat_cluster_n2
5401 <1>
5402 <1> loc_createfile_dsk_wr_err:
5403 <1> ; 16/10/2016 (ldh -> 18)
5404 <1> ; 23/03/2016
5405 <1> ;mov     eax, 18 ; Drive not ready or write error !
5406 <1> ; 29/07/2022
5407 0000BE46 29C0 <1> sub     eax, eax
5408 0000BE48 B012 <1> mov     al, 18
5409 <1> loc_cf_stc_retn:
5410 0000BE4A E9BA000000 <1> jmp     loc_createfile_stc_retn
5411 <1>
5412 <1> loc_createfile_update_fat_cluster_n1:
5413 0000BE4F A1[C0820100] <1> mov     eax, [createfile_PCluster]
5414 0000BE54 8B0D[BC820100] <1> mov     ecx, [createfile_Cluster]
5415 0000BE5A E801040000 <1> call    update_cluster
5416 0000BE5F 7304 <1> jnc     short loc_createfile_update_fat_cluster_n2
5417 0000BE61 09C0 <1> or      eax, eax ; EAX = 0 -> cluster value is 0 or eocc
5418 <1> ;jnz     loc_createfile_stc_retn
5419 <1> ; 29/07/2022
5420 0000BE63 75E5 <1> jnz     short loc_cf_stc_retn
5421 <1>
5422 <1> loc_createfile_update_fat_cluster_n2:
5423 0000BE65 A1[BC820100] <1> mov     eax, [createfile_Cluster]
5424 <1> ;mov     ecx, 0FFFFFFFh
5425 <1> ; 29/08/2024
5426 <1> ; (update_cluster will save 28 bit cluster value)
5427 <1> ;mov     ecx, 0FFFFFFFh
5428 0000BE6A 29C9 <1> sub     ecx, ecx
5429 0000BE6C 49 <1> dec     ecx
5430 0000BE6D E8EE030000 <1> call    update_cluster
5431 0000BE72 7353 <1> jnc     short loc_createfile_save_fat_buffer_3
5432 0000BE74 09C0 <1> or      eax, eax ; EAX = 0 -> cluster value is 0 or eocc
5433 0000BE76 744F <1> jz      short loc_createfile_save_fat_buffer_3
5434 <1>
5435 <1> loc_cf_upd_fat_fcluster_stc_retn:
5436 0000BE78 E98C000000 <1> jmp     loc_createfile_stc_retn
5437 <1>
5438 <1> loc_createfile_get_set_wfc_cont:

```

```

5439 <1> ;movzx ecx, word [esi+LD_BPB+BytesPerSec] ; 512
5440 <1> ;movzx ecx, word [createfile_BytesPerSec] ; 512
5441 <1> ; 19/12/2025
5442 0000BE7D B900020000 <1> mov ecx, 512
5443 0000BE82 01C8 <1> add eax, ecx
5444 0000BE84 48 <1> dec eax ; add eax, 511
5445 0000BE85 29D2 <1> sub edx, edx
5446 0000BE87 F7F1 <1> div ecx
5447 0000BE89 0FB61D[C5820100] <1> movzx ebx, byte [createfile_SecPerClust]
5448 0000BE90 01D8 <1> add eax, ebx
5449 0000BE92 48 <1> dec eax ; add eax, SecPerClust - 1
5450 <1> ;xor dx, dx
5451 <1> ; 27/08/2024
5452 0000BE93 31D2 <1> xor edx, edx
5453 0000BE95 F7F3 <1> div ebx
5454 <1>
5455 <1> loc_createfile_set_cluster_count:
5456 0000BE97 A3[C8820100] <1> mov [createfile_CCount], eax
5457 <1>
5458 <1> ; 31/08/2024
5459 <1> ;;;
5460 0000BE9C 833D[B0820100]00 <1> cmp dword [createfile_size], 0 ; empty file ?
5461 0000BEA3 7422 <1> jz short loc_createfile_save_fat_buffer_3
5462 <1> ;;;
5463 <1>
5464 0000BEA5 BF00000700 <1> mov edi, Cluster_Buffer
5465 0000BEAA 89C8 <1> mov eax, ecx ; Bytes per Sector
5466 0000BEAC F7E3 <1> mul ebx ; Sectors per Cluster
5467 <1> ; EAX = Bytes per Cluster
5468 0000BEAE 89C1 <1> mov ecx, eax
5469 0000BEB0 C1E902 <1> shr ecx, 2 ; dword count
5470 0000BEB3 31C0 <1> xor eax, eax
5471 0000BEB5 F3AB <1> rep stosd ; clear cluster buffer
5472 <1>
5473 0000BEB7 A1[BC820100] <1> mov eax, [createfile_cluster] ; 24/03/2016
5474 <1>
5475 0000BEBE 89D9 <1> mov ecx, ebx
5476 <1>
5477 <1> loc_createfile_get_set_wf_fclust_cont:
5478 <1> ;sub eax, 2
5479 <1> ; 30/07/2022
5480 0000BEBE 48 <1> dec eax
5481 0000BEBF 48 <1> dec eax
5482 0000BEC0 F7E1 <1> mul ecx
5483 <1> ; EAX = Logical DOS disk address (offset)
5484 0000BEC2 E95FFFFFFF <1> jmp loc_createfile_write_fcluster_to_disk
5485 <1>
5486 <1> loc_createfile_save_fat_buffer_3:
5487 <1> ; byte [FAT_BuffValidData] = 2
5488 0000BEC7 E808060000 <1> call save_fat_buffer
5489 <1> ;jc loc_createfile_stc_retn
5490 <1> ; 29/07/2022
5491 0000BECC 72AA <1> jc short loc_cf_upd_fat_fcluster_stc_retn
5492 <1>
5493 <1> ; 21/03/2016
5494 <1> ;cmp byte [FAT_ClusterCounter], 1
5495 <1> ;jb short loc_createfile_save_fat_buffer_4
5496 <1> ; 31/08/2024
5497 0000BECE 803D[7E7E0100]00 <1> cmp byte [FAT_ClusterCounter], 0
5498 0000BED5 761B <1> jna short loc_createfile_save_fat_buffer_4 ; cf = 0
5499 <1>
5500 <1> ; ESI = Logical DOS Drive Description Table address
5501 0000BED7 A1[7E7E0100] <1> mov eax, [FAT_ClusterCounter]
5502 0000BEDC 66BB01FF <1> mov bx, 0FF01h ; add free clusters
5503 0000BEE0 E87C060000 <1> call calculate_fat_freespace
5504 <1>
5505 <1> ;inc eax ; 0FFFFFFFh -> 0 ; recalculation is needed!
5506 <1> ;jnz short loc_createfile_save_fat_buffer_4
5507 <1>
5508 <1> ; ecx > 0 -> Recalculation is needed
5509 0000BEE5 09C9 <1> or ecx, ecx
5510 0000BEE7 7409 <1> jz short loc_createfile_save_fat_buffer_4
5511 <1>
5512 0000BEE9 66BB00FF <1> mov bx, 0FF00h ; ; recalculate free space
5513 0000BEED E86F060000 <1> call calculate_fat_freespace
5514 <1>
5515 <1> loc_createfile_save_fat_buffer_4:
5516 0000BEF2 FF0D[C8820100] <1> dec dword [createfile_CCount]
5517 <1> ;jz short loc_createfile_upd_dir_modif_date_time
5518 0000BEF8 743D <1> jz short loc_createfile_stc_retn_cc ; 31/03/2016
5519 <1>
5520 <1> loc_createfile_get_set_write_next_cluster:
5521 0000BEFA E88C020000 <1> call get_first_free_cluster
5522 0000BEFF 7208 <1> jc short loc_createfile_stc_retn
5523 <1>
5524 <1> loc_createfile_get_set_write_next_cluster_1:
5525 0000BF01 83F8FF <1> cmp eax, 0FFFFFFFh
5526 0000BF04 7211 <1> jb short loc_createfile_get_set_write_next_cluster_2
5527 <1>
5528 <1> ;loc_createfile_wnc_insufficient_disk_space:
5529 <1> ;mov eax, 27h ; Insufficient disk space
5530 <1> ; 29/07/2022
5531 <1> ;xor eax, eax
5532 <1> ; 29/08/2024
5533 <1> ; eax = 0FFFFFFFh
5534 0000BF06 40 <1> inc eax ; eax = 0
5535 0000BF07 B027 <1> mov al, 27h
5536 <1>
5537 <1> loc_createfile_stc_retn:
5538 0000BF09 803D[CC820100]01 <1> cmp byte [createfile_wfc], 1
5539 0000BF10 7324 <1> jnb short loc_createfile_err_retn
5540 0000BF12 C3 <1> retn
5541 <1>
5542 <1> loc_createfile_wnc_inv_format_retn:
5543 <1> ;mov eax, 28
5544 0000BF13 B01C <1> mov al, 28 ; Invalid format
5545 0000BF15 EBF2 <1> jmp short loc_createfile_stc_retn
5546 <1>
5547 <1> loc_createfile_get_set_write_next_cluster_2:
5548 0000BF17 83F802 <1> cmp eax, 2
5549 0000BF1A 72F7 <1> jb short loc_createfile_wnc_inv_format_retn
5550 <1>
5551 <1> loc_createfile_get_set_write_next_cluster_3:
5552 0000BF1C 8B0D[BC820100] <1> mov ecx, [createfile_cluster]
5553 0000BF22 A3[BC820100] <1> mov [createfile_cluster], eax
5554 0000BF27 890D[C0820100] <1> mov [createfile_PCluster], ecx
5555 0000BF2D 0FB60D[C5820100] <1> movzx ecx, byte [createfile_SecPerClust]
5556 0000BF34 E888 <1> jmp short loc_createfile_get_set_wf_fclust_cont
5557 <1>
5558 <1> loc_createfile_err_retn:
5559 0000BF36 F9 <1> stc
5560 <1>
5561 <1> ;loc_createfile_upd_dir_modif_date_time:
5562 <1> loc_createfile_stc_retn_cc: ; 31/03/2016

```

```

5563 0000BF37 9C      <1>      pushf    ; cpu is here for an error return or completion
5564 0000BF38 50      <1>      push     eax ; error code if cf = 1
5565                  <1>
5566                  <1>      ;call    update_parent_dir_lmdt
5567                  <1>
5568                  <1> ;loc_createfile_stc_retn_cc:
5569 0000BF39 A1[7E7E0100] <1>      mov     eax, [FAT_ClusterCounter]
5570 0000BF3E 09C0      <1>      or      eax, eax
5571 0000BF40 741A      <1>      jz      short loc_createfile_stc_retn_pop_eax
5572 0000BF42 8A3D[62770100] <1>      mov     bh, [Current_Drv]
5573 0000BF48 B301      <1>      mov     bl, 01h ; BL = 1 -> add clusters
5574                  <1>      ; NOTE: EAX value will be added to Free Cluster Count
5575                  <1>      ; (If EAX value is negative, Free Cluster Count will be decreased)
5576 0000BF4A E812060000 <1>      call    calculate_fat_freespace
5577                  <1>      ; ESI = Logical DOS Drive Description Table Address
5578                  <1>      ;jc short loc_createfile_stc_retn_pop_eax_cf
5579 0000BF4F 21C9      <1>      and     ecx, ecx ; cx = 0 -> valid free sector count
5580 0000BF51 7409      <1>      jz      short loc_createfile_stc_retn_pop_eax
5581                  <1>
5582                  <1> loc_createfile_stc_retn_recalc_FAT_freespace:
5583 0000BF53 66BB00FF <1>      mov     bx, 0FF00h ; bh = 0FFh ->
5584                  <1>      ; ESI = Logical DOS Drv DT Addr
5585                  <1>      ; BL = 0 -> Recalculate
5586 0000BF57 E805060000 <1>      call    calculate_fat_freespace
5587                  <1>
5588                  <1> loc_createfile_stc_retn_pop_eax:
5589 0000BF5C 58          <1>      pop     eax
5590 0000BF5D 9D          <1>      popf
5591 0000BF5E 7218      <1>      jc      short loc_createfile_retn
5592                  <1>
5593                  <1> loc_createfile_retn_fcluster:
5594 0000BF60 A1[B4820100] <1>      mov     eax, [createfile_FFcluster]
5595                  <1>      ;mov     ebx, createfile_size
5596                  <1>      ;movzx ecx, byte [esi+LD_BP8+SecPerClust]
5597                  <1>      ;movzx ecx, byte [createfile_SecPerClust] ; 23/03/2016
5598                  <1>      ;movzx edx, word [createfile_DirIndex]
5599                  <1>      ; 29/08/2024 - TRDOS 386 v2.0.9
5600 0000BF65 8B1D[B0820100] <1>      mov     ebx, [createfile_size]
5601 0000BF6B 0FB70D[C6820100] <1>      movzx   ecx, word [createfile_DirIndex] ; = directory entry index
5602 0000BF72 8B15[B8820100] <1>      mov     edx, [createfile_LastDirCluster] ; = [DirBuff_Cluster]
5603                  <1>      ; esi = Logical DOS Drive Description Table address
5604                  <1>      ; byte [esi] = drive name ; 'A','B','C','D'
5605                  <1>
5606                  <1> loc_createfile_retn:
5607 0000BF78 C3          <1>      retn
5608                  <1>
5609                  <1> ; 19/12/2025 (TRDOS 386 Kernel v2.0.10)
5610                  <1> ; 28/07/2022 (TRDOS 386 Kernel v2.0.5)
5611                  <1>
5612                  <1> ;create_fs_file:
5613                  <1>      ; temporary (21/03/2016)
5614                  <1>      ;retn
5615                  <1>
5616                  <1> ;delete_fs_file:
5617                  <1>      ; temporary (28/02/2016)
5618                  <1>      ;retn
5619                  <1>
5620                  <1> ;rename_fs_file_or_directory:
5621                  <1>      ;retn
5622                  <1>
5623                  <1> ;make_fs_directory:
5624                  <1>      ; temporary (21/02/2016)
5625                  <1>      ;retn
5626                  <1>
5627                  <1> ;add_new_fs_section:
5628                  <1>      ; temporary (11/03/2016)
5629                  <1>      ;retn
5630                  <1>
5631                  <1> ;delete_fs_directory_entry:
5632                  <1>      ; temporary (11/03/2016)
5633                  <1>      ;retn
5634                  <1>
5635                  <1> ;csftdf2_read_fs_file_sectors:
5636                  <1>      ; temporary (19/03/2016)
5637                  <1>      ;retn
5638                  <1>
5639                  <1> ;csftdf2_write_fs_file_sectors:
5640                  <1>      ; temporary (19/03/2016)
5641                  <1>      ;retn
5642                  <1>
5643                  <1> %include 'trdosk5.s' ; 24/01/2016
5644                  <1> *****
5645                  <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.10) - File System Procedures : trdosk5s
5646                  <1>
5647                  <1> ; Last Update: 21/12/2025 (Previous: 31/08/2024, v2.0.9)
5648                  <1> -----
5649                  <1> ; Beginning: 24/01/2016
5650                  <1> -----
5651                  <1> ; Assembler: NASM version 2.11 (trdos386.s)
5652                  <1> -----
5653                  <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
5654                  <1> ; DRV_FAT.ASM (21/08/2011)
5655                  <1> ; *****
5656                  <1> ; DRV_FAT.ASM (c) 2005-2011 Erdogan TAN [ 07/07/2009 ] Last Update: 21/08/2011
5657                  <1>
5658                  <1> get_next_cluster:
5659                  <1>      ; 07/08/2022
5660                  <1>      ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
5661                  <1>      ; 15/10/2016
5662                  <1>      ; 23/03/2016
5663                  <1>      ; 01/02/2016 (TRDOS 386 = TRDOS v2.0)
5664                  <1>      ; 05/07/2011
5665                  <1>      ; 07/07/2009
5666                  <1>      ; 2005
5667                  <1>      ; INPUT ->
5668                  <1>      ; EAX = Cluster Number (32 bit)
5669                  <1>      ; ESI = Logical DOS Drive Parameters Table
5670                  <1>      ; OUTPUT ->
5671                  <1>      ; cf = 0 -> No Error, EAX valid
5672                  <1>      ; cf = 1 & EAX = 0 -> End of Cluster Chain
5673                  <1>      ; cf = 1 & EAX > 0 -> Error
5674                  <1>      ; ECX = Current/Previous cluster (if CF = 0)
5675                  <1>      ; EAX = Next Cluster Number (32 bit)
5676                  <1>
5677                  <1>      ; (Modified registers: EAX, ECX, EBX, EDX)
5678                  <1>
5679                  <1> mov     [FAT_CurrentCluster], eax
5680                  <1> check_next_cluster_fat_type:
5681                  <1>      sub     edx, edx ; 0
5682                  <1>      cmp     byte [esi+LD_FATType], 2
5683                  <1>      jnb     short get_FAT12_next_cluster
5684                  <1>      ;ja     get_FAT32_next_cluster
5685                  <1>      ; 25/07/2022
5686                  <1>      jna     short get_FAT16_next_cluster
5687                  <1>      jmp     get_FAT32_next_cluster

```

```

45 <1>
46 <1> get_FAT16_next_cluster:
47 0000BF8D BB00030000 <1> mov ebx, 300h ;768
48 0000BF92 F7F3 <1> div ebx
49 <1> ; EAX = Count of 3 FAT sectors
50 <1> ; EDX = Cluster Offset (< 768)
51 <1> ;shl dx, 1 ; Multiply by 2
52 <1> ; 25/07/2022
53 0000BF94 D1E2 <1> shl edx, 1
54 0000BF96 89D3 <1> mov ebx, edx ; Byte Offset
55 0000BF98 81C3001C0900 <1> add ebx, FAT_Buffer
56 0000BF9E 66BA0300 <1> mov dx, 3
57 0000BFA2 F7E2 <1> mul edx
58 <1> ; EAX = FAT Sector (<= 256)
59 <1> ; EDX = 0
60 0000BFA4 8A0E <1> mov cl, [esi+LD_Name]
61 <1> ;cmp byte [FAT_BuffValidData], 0
62 0000BFA6 3815[767E0100] <1> cmp [FAT_BuffValidData], dl ; 0
63 0000BFAC 7674 <1> jna short load_FAT_sectors0
64 0000BFAE 3A0D[777E0100] <1> cmp cl, [FAT_BuffDrvName]
65 0000BFB4 756C <1> jne short load_FAT_sectors0
66 0000BFB6 3B05[7A7E0100] <1> cmp eax, [FAT_BuffSector]
67 0000BFB8 756A <1> jne short load_FAT_sectors1
68 <1> ;movzx eax, word [ebx]
69 0000BFBE 668B03 <1> mov ax, [ebx]
70 <1> ; 01/02/2016
71 <1> ; DRV_FAT.ASM (21/08/2011) had a FAT16 bug here !
72 <1> ; (cmp ah, 0Fh) ! (ax >= FF7h)
73 <1> ; (how can i do a such mistake!?)
74 <1> ;cmp al, 0F7h
75 <1> ;jb short loc_pass_gnc_FAT16_eoc_check
76 <1> ;cmp ah, 0FFh
77 <1> ;jb short loc_pass_gnc_FAT16_eoc_check
78 0000BFC1 6683F8F7 <1> cmp ax, 0FFF7h
79 0000BFC5 724E <1> jb short loc_pass_gnc_FAT16_eoc_check
80 <1> ; ax >= 0FFF7h (cluster 0002h to 0FFF6h is valid, in use)
81 0000BFC7 EB4A <1> jmp short loc_pass_gnc_FAT16_eoc_check_xor_eax
82 <1>
83 <1> get_FAT12_next_cluster:
84 0000BFC9 BB00040000 <1> mov ebx, 400h ;1024
85 0000BFCE F7F3 <1> div ebx
86 <1> ; EAX = Count of 3 FAT sectors
87 <1> ; EDX = Cluster Offset (< 1024)
88 <1> ; 25/07/2022
89 <1> ;push ax
90 0000BFD0 50 <1> push eax
91 <1> ;mov ax, 3
92 0000BFD1 B003 <1> mov al, 3
93 <1> ;mul dx ; Multiply by 3
94 0000BFD3 F7E2 <1> mul edx
95 <1> ;shr ax, 1 ; Divide by 2
96 0000BFD5 D1E8 <1> shr eax, 1
97 <1> ;mov bx, ax ; Byte Offset
98 0000BFD7 89C3 <1> mov ebx, eax
99 0000BFD9 81C3001C0900 <1> add ebx, FAT_Buffer
100 0000BFD5 58 <1> pop eax
101 <1> ;pop ax
102 <1> ;mov dx, 3
103 0000BFED B203 <1> mov dl, 3
104 0000BFE2 F7E2 <1> mul edx
105 <1> ; EAX = FAT Sector (<= 12)
106 <1> ; EDX = 0
107 0000BFE4 8A0E <1> mov cl, [esi+LD_Name]
108 <1> ;cmp byte [FAT_BuffValidData], 0
109 0000BFE6 3815[767E0100] <1> cmp [FAT_BuffValidData], dl ; 0
110 0000BFEC 7634 <1> jna short load_FAT_sectors0
111 0000BFEE 3A0D[777E0100] <1> cmp cl, [FAT_BuffDrvName]
112 0000BFF4 752C <1> jne short load_FAT_sectors0
113 0000BFF6 3B05[7A7E0100] <1> cmp eax, [FAT_BuffSector]
114 0000BFFC 752A <1> jne short load_FAT_sectors1
115 0000BFFE A1[727E0100] <1> mov eax, [FAT_CurrentCluster]
116 <1> ;shr ax, 1
117 <1> ; 25/07/2022
118 0000C003 D1E8 <1> shr eax, 1
119 <1> ;movzx eax, word [ebx]
120 0000C005 668B03 <1> mov ax, [ebx]
121 0000C008 7313 <1> jnc short get_FAT12_nc_even
122 <1> ;shr ax, 4
123 0000C00A C1E804 <1> shr eax, 4
124 <1> loc_gnc_fat12_eoc_check:
125 <1> ;cmp al, 0F7h
126 <1> ;jb short loc_pass_gnc_FAT16_eoc_check
127 <1> ;cmp ah, 0Fh
128 <1> ;jb short loc_pass_gnc_FAT16_eoc_check
129 0000C00D 663DF70F <1> cmp ax, 0FFF7h
130 0000C011 7202 <1> jb short loc_pass_gnc_FAT16_eoc_check
131 <1> ; ax >= 0FFF7h (cluster 0002h to 0FFF6h is valid, in use)
132 <1>
133 <1> loc_pass_gnc_FAT16_eoc_check_xor_eax:
134 0000C013 31C0 <1> xor eax, eax ; 0
135 <1> loc_pass_gnc_FAT16_eoc_check:
136 <1> loc_pass_gnc_FAT32_eoc_check:
137 0000C015 8B0D[727E0100] <1> mov ecx, [FAT_CurrentCluster]
138 0000C01B F5 <1> cmc
139 0000C01C C3 <1> retn
140 <1>
141 <1> get_FAT12_nc_even:
142 0000C01D 80E40F <1> and ah, 0Fh
143 0000C020 EBEB <1> jmp short loc_gnc_fat12_eoc_check
144 <1>
145 <1> load_FAT_sectors0:
146 0000C022 880D[777E0100] <1> mov [FAT_BuffDrvName], cl
147 <1> load_FAT_sectors1:
148 <1> ; 25/07/2022
149 <1> ;sub edx, edx
150 <1> ; edx = 0
151 0000C028 A3[7A7E0100] <1> mov [FAT_BuffSector], eax
152 0000C02D 89C3 <1> mov ebx, eax
153 0000C02F 034660 <1> add eax, [esi+LD_FATBegin]
154 0000C032 B202 <1> mov dl, 2
155 <1> ;cmp byte [esi+LD_FATType], 2
156 0000C034 385603 <1> cmp [esi+LD_FATType], dl ; 2
157 0000C037 7748 <1> ja short load_FAT_sectors3
158 0000C039 0FB74E1C <1> movzx ecx, word [esi+LD_BP8+BPB_FATSz16]
159 0000C03D EB45 <1> jmp short load_FAT_sectors4
160 <1>
161 <1> ; 07/08/2022
162 <1> get_FAT32_next_cluster:
163 0000C03F BB80010000 <1> mov ebx, 180h ;384
164 0000C044 F7F3 <1> div ebx
165 <1> ; EAX = Count of 3 FAT sectors
166 <1> ; EDX = Cluster Offset (< 384)
167 <1> ;shl dx, 2 ; Multiply by 4
168 <1> ; 25/07/2022

```

```

169 0000C046 C1E202      <1>      shl     edx, 2
170 0000C049 89D3      <1>      mov     ebx, edx ; Byte Offset
171 0000C04B 81C3001C0900      <1>      add     ebx, FAT_Buffer
172 0000C051 66BA0300      <1>      mov     dx, 3
173 0000C055 F7E2      <1>      mul     edx
174      <1>      ; EAX = FAT Sector (<= 2097152) ; (FFFFFF7h * 4) / 512
175      <1>      ; for 32KB cluster size:
176      <1>      ; EAX <= 1024 = (4GB / 32KB) * 4) / 512
177      <1>      ; EDX = 0
178 0000C057 8A0E      <1>      mov     cl, [esi+LD_Name]
179      <1>      ;cmp    byte [FAT_BuffValidData], 0
180 0000C059 3815[767E0100]      <1>      cmp     [FAT_BuffValidData], dl ; 0
181 0000C05F 76C1      <1>      jna     short load_FAT_sectors0
182 0000C061 3A0D[777E0100]      <1>      cmp     cl, [FAT_BuffDrvName]
183 0000C067 75B9      <1>      jne     short load_FAT_sectors0
184 0000C069 3B05[7A7E0100]      <1>      cmp     eax, [FAT_BuffSector] ; 0, 3, 6, 9 ...
185 0000C06F 75B7      <1>      jne     short load_FAT_sectors1
186 0000C071 8B03      <1>      mov     eax, [ebx]
187 0000C073 25FFFFFF0F      <1>      and     eax, 0FFFFFFh ; 28 bit Cluster
188 0000C078 3DF7FFFF0F      <1>      cmp     eax, 0FFFFFF7h
189 0000C07D 7296      <1>      jnb     short loc_pass_gnc_FAT32_eoc_check
190      <1>      ; eax >= FFFFFFF7h (cluster 0002h to FFFFFFF6h is valid)
191 0000C07F EB92      <1>      jmp     short loc_pass_gnc_FAT16_eoc_check_xor_eax
192      <1>
193      <1> load_FAT_sectors3:
194 0000C081 8B4E2A      <1>      mov     ecx, [esi+LD_BPB+BPB_FATSz32]
195      <1> load_FAT_sectors4:
196 0000C084 29D9      <1>      sub     ecx, ebx ; [FAT_BuffSector]
197      <1>      ; 25/07/2022
198 0000C086 FEC2      <1>      inc     dl
199      <1>      ; edx = 3
200      <1>      ;cmp    ecx, 3
201 0000C088 39D1      <1>      cmp     ecx, edx ; 3
202 0000C08A 7602      <1>      jna     short load_FAT_sectors5
203      <1>      ;mov    ecx, 3
204      <1>      ; 25/07/2022
205 0000C08C 89D1      <1>      mov     ecx, edx ; 3
206      <1> load_FAT_sectors5:
207 0000C08E BB001C0900      <1>      mov     ebx, FAT_Buffer
208 0000C093 E84B590000      <1>      call    disk_read
209 0000C098 730C      <1>      jnc     short load_FAT_sectors_ok
210      <1>      ; 15/10/2016 (15h -> 17)
211      <1>      ; 23/03/2016 (15h)
212 0000C09A B811000000      <1>      mov     eax, 17 ; Drive not ready or read error
213      <1>      ;mov    byte [FAT_BuffValidData], 0
214      <1>      ; 25/07/2022
215 0000C09F 8825[767E0100]      <1>      mov     byte [FAT_BuffValidData], ah ; 0
216 0000C0A5 C3      <1>      retn
217      <1> load_FAT_sectors_ok:
218 0000C0A6 C605[767E0100]01      <1>      mov     byte [FAT_BuffValidData], 1
219 0000C0AD A1[727E0100]      <1>      mov     eax, [FAT_CurrentCluster]
220 0000C0B2 E9C7FEFFFF      <1>      jmp     check_next_cluster_fat_type
221      <1>
222      <1> load_FAT_root_directory:
223      <1>      ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
224      <1>      ; 23/10/2016
225      <1>      ; 15/10/2016
226      <1>      ; 07/02/2016
227      <1>      ; 02/02/2016
228      <1>      ; 01/02/2016 (TRDOS 386 = TRDOS v2.0)
229      <1>      ; 21/05/2011
230      <1>      ; 22/08/2009
231      <1>      ;
232      <1>      ; INPUT ->
233      <1>      ; ESI = Logical DOS Drive Description Table
234      <1>      ; OUTPUT ->
235      <1>      ; cf = 1 -> Root directory could not be loaded
236      <1>      ; EAX > 0 -> Error number
237      <1>      ; cf = 0 -> EAX = 0
238      <1>      ; ECX = Directory buffer size in sectors (CL)
239      <1>      ; EBX = Directory buffer address
240      <1>      ; NOTE: DirBuffer_Size is in bytes ! (word)
241      <1>      ;
242      <1>      ; (Modified registers: EAX, ECX, EBX, EDX)
243      <1>      ;
244      <1>      ; NOTE: Only for FAT12 and FAT16 file systems !
245      <1>      ; (FAT32 fs root dir must be loaded as sub directory)
246      <1>
247 0000C0B7 8A1E      <1>      mov     bl, [esi+LD_Name]
248 0000C0B9 8A7E03      <1>      mov     bh, [esi+LD_FATType]
249      <1>
250      <1>      ;mov    [DirBuff_DRV], bl
251      <1>      ;mov    [DirBuff_FATType], bh
252 0000C0BC 66891D[877E0100]      <1>      mov     [DirBuff_DRV], bx
253      <1>
254      <1>      ;cmp    bh, 2
255      <1>      ;ja     short load_FAT32_root_dir0 ; FAT32 root dir
256      <1>
257      <1> load_FAT_root_dir0: ; 23/10/2016
258 0000C0C3 0FB75617      <1>      movzx   edx, word [esi+LD_BPB+RootDirEnts]
259      <1>
260      <1>      ;or     dx, dx ; 0 for FAT32 file systems
261      <1>      ;jz     short load_FAT32_root_dir0 ; FAT32 root dir
262      <1>
263      <1>      ; 25/07/2022
264 0000C0C7 89D0      <1>      mov     eax, edx
265 0000C0C9 6681FA0002      <1>      cmp     dx, 512 ; Number of Root Dir Entries
266 0000C0CE 740B      <1>      je      short lrd_mov_ecx_32
267      <1>      ;mov    eax, edx ; 25/07/2022
268      <1>      ; 23/10/2016
269 0000C0D0 89C1      <1>      mov     ecx, eax
270 0000C0D2 6683C10F      <1>      add     cx, 15 ; round up
271      <1>      ;shr    cx, 4 ; 16 entries per sector (512/32)
272      <1>      ; 25/07/2022
273 0000C0D6 C1E904      <1>      shr     ecx, 4
274      <1>      ; ecx = Root directory size in sectors
275      <1>      ;shl    ax, 5 ; Root directory size in bytes
276      <1>      ; 25/07/2022
277      <1>      ;shl    eax, 5
278      <1>      ;dec    dx ; Last entry number of root dir
279      <1>      ;dec    edx
280      <1>      ; cx = Dir Buffer sector count
281 0000C0D9 EB04      <1>      jmp     short lrd_check_dir_buffer
282      <1>
283      <1> lrd_mov_ecx_32:
284      <1>      ;mov    ecx, 32
285      <1>      ; 25/07/2022
286 0000C0DB 29C9      <1>      sub     ecx, ecx
287 0000C0DD B120      <1>      mov     cl, 32
288      <1>      ;dec    dx ; 511
289      <1>      ;mov    ax, 32*512
290      <1>
291      <1> lrd_check_dir_buffer:
292      <1>      ; 25/07/2022

```

```

293 0000C0DF 4A          <1>    dec     edx ; root dir entries - 1
294 0000C0E0 C1E005     <1>    shl     eax, 5 ; * 32
295                      <1>    ;
296 0000C0E3 29DB       <1>    sub     ebx, ebx ; 0
297 0000C0E5 881D[897E0100] <1>    mov     [DirBuff_ValidData], bl ; 0
298 0000C0EB 668915[8C7E0100] <1>    mov     [DirBuff_LastEntry], dx
299 0000C0F2 891D[8E7E0100] <1>    mov     [DirBuff_Cluster], ebx ; 0
300 0000C0F8 66A3[927E0100] <1>    mov     [DirBuffer_Size], ax
301                      <1>    ;
302 0000C0FE 8B4664     <1>    mov     eax, [esi+LD_ROOTBegin]
303                      <1>    read_directory:
304 0000C101 BB00000800     <1>    mov     ebx, Directory_Buffer
305 0000C106 51          <1>    push    ecx ; Directory buffer sector count
306 0000C107 53          <1>    push    ebx
307 0000C108 E8D6580000 <1>    call    disk_read
308 0000C10D 5B          <1>    pop     ebx
309 0000C10E 720B       <1>    jc      short load_DirBuff_error
310                      <1>    ;
311                      <1>    validate_DirBuff_and_return:
312 0000C110 59          <1>    pop     ecx ; Number of loaded sectors
313 0000C111 C605[897E0100]01 <1>    mov     byte [DirBuff_ValidData], 1
314 0000C118 31C0       <1>    xor     eax, eax ; 0 = no error
315 0000C11A C3          <1>    retn
316                      <1>    ;
317                      <1>    load_DirBuff_error:
318 0000C11B 89C8       <1>    mov     eax, ecx ; remaining sectors
319 0000C11D 59          <1>    pop     ecx ; sector count
320 0000C11E 29C1       <1>    sub     ecx, eax ; Number of loaded sectors
321                      <1>    ; 15/10/2016 (15h -> 17)
322                      <1>    ; mov     eax, 17 ; DRV NOT READY OR READ ERROR !
323                      <1>    ; 25/07/2022
324                      <1>    ; sub     eax, eax
325 0000C120 B011       <1>    mov     al, 17
326 0000C122 F9          <1>    stc
327 0000C123 C3          <1>    retn
328                      <1>    ;
329                      <1>    load_FAT32_root_directory:
330                      <1>    ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
331                      <1>    ; 02/02/2016 (TRDOS 386 = TRDOS v2.0)
332                      <1>    ;
333                      <1>    ; INPUT ->
334                      <1>    ; ESI = Logical DOS Drive Description Table
335                      <1>    ; OUTPUT ->
336                      <1>    ; cf = 1 -> Root directory could not be loaded
337                      <1>    ; EAX > 0 -> Error number
338                      <1>    ; cf = 0 -> EAX = 0
339                      <1>    ; ECX = Directory buffer size in sectors (CL)
340                      <1>    ; EBX = Directory buffer address
341                      <1>    ; NOTE: DirBuffer_Size is in bytes ! (word)
342                      <1>    ;
343                      <1>    ; (Modified registers: EAX, ECX, EBX, EDX)
344                      <1>    ;
345 0000C124 8A1E       <1>    mov     bl, [esi+LD_Name]
346 0000C126 8A7E03     <1>    mov     bh, [esi+LD_FATType]
347                      <1>    ;
348                      <1>    ; mov     [DirBuff_DRV], bl
349                      <1>    ; mov     [DirBuff_FATType], bh
350 0000C129 66891D[877E0100] <1>    mov     [DirBuff_DRV], bx
351                      <1>    ;
352                      <1>    load_FAT32_root_dir0:
353 0000C130 8B4632     <1>    mov     eax, [esi+LD_BPB+FAT32_RootFClust]
354 0000C133 EB0C       <1>    jmp     short load_FAT_sub_dir0
355                      <1>    ;
356                      <1>    load_FAT_sub_directory:
357                      <1>    ; 19/12/2025 (TRDOS 386 Kernel v2.0.10)
358                      <1>    ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
359                      <1>    ; 01/02/2016 (TRDOS 386 = TRDOS v2.0)
360                      <1>    ; 05/07/2011
361                      <1>    ; 23/08/2009
362                      <1>    ;
363                      <1>    ; INPUT ->
364                      <1>    ; ESI = Logical DOS Drive Description Table
365                      <1>    ; EAX = Cluster Number
366                      <1>    ; OUTPUT ->
367                      <1>    ; cf = 1 -> Sub directory could not be loaded
368                      <1>    ; EAX > 0 -> Error number
369                      <1>    ; cf = 0 -> EAX = 0
370                      <1>    ; ECX = Directory buffer size in sectors (CL)
371                      <1>    ; EBX = Directory buffer address
372                      <1>    ;
373                      <1>    ; NOTE: DirBuffer_Size is in bytes ! (word)
374                      <1>    ;
375                      <1>    ; (Modified registers: EAX, ECX, EBX, EDX)
376                      <1>    ;
377 0000C135 8A1E       <1>    mov     bl, [esi+LD_Name]
378 0000C137 8A7E03     <1>    mov     bh, [esi+LD_FATType]
379                      <1>    ;
380                      <1>    ; mov     [DirBuff_DRV], bl
381                      <1>    ; mov     [DirBuff_FATType], bh
382 0000C13A 66891D[877E0100] <1>    mov     [DirBuff_DRV], bx
383                      <1>    ;
384                      <1>    load_FAT_sub_dir0:
385 0000C141 0FB64E13     <1>    movzx   ecx, byte [esi+LD_BPB+SecPerClust]
386                      <1>    ;
387 0000C145 882D[897E0100] <1>    mov     [DirBuff_ValidData], ch ; 0
388 0000C14B A3[8E7E0100] <1>    mov     [DirBuff_Cluster], eax
389                      <1>    ;
390                      <1>    ; movzx   eax, word [esi+LD_BPB+BytesPerSec] ; 512
391                      <1>    ; mul     ecx
392                      <1>    ; shr     eax, 5 ; directory entry count (dir size / 32)
393                      <1>    ; 19/12/2025
394 0000C150 89C8       <1>    mov     eax, ecx
395 0000C152 C1E004     <1>    shl     eax, 4 ; 512/32 (16 entries per sector)
396                      <1>    ;
397                      <1>    ; dec     ax ; last entry
398                      <1>    ; 25/07/2022
399 0000C155 48          <1>    dec     eax
400 0000C156 66A3[8C7E0100] <1>    mov     [DirBuff_LastEntry], ax
401                      <1>    ;
402 0000C15C A1[8E7E0100] <1>    mov     eax, [DirBuff_Cluster]
403 0000C161 83E802     <1>    sub     eax, 2
404 0000C164 F7E1       <1>    mul     ecx
405 0000C166 034668     <1>    add     eax, [esi+LD_DATABegin]
406                      <1>    ; ecx = sectors per cluster (dir buffer size <= 128 sectors)
407 0000C169 EB96       <1>    jmp     short read_directory
408                      <1>    ;
409                      <1>    ; DRV_FS.ASM
410                      <1>    ;
411                      <1>    ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
412                      <1>    ;
413                      <1>    ; 19/12/2025
414                      <1>    ; load_current_FS_directory:
415                      <1>    ; retn
416                      <1>    ; load_FS_root_directory:

```



```

417 <1> ; ;retn
418 <1> ;load_FS_sub_directory:
419 <1> ; ;retn
420 <1>
421 <1> read_cluster:
422 <1> ; 19/12/2025 (TRDOS 386 v2.0.10)
423 <1> ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
424 <1> ; 15/10/2016
425 <1> ; 18/03/2016
426 <1> ; 16/03/2016
427 <1> ; 17/02/2016
428 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
429 <1> ;
430 <1> ; INPUT ->
431 <1> ; EAX = Cluster Number (Sector index for SINGLIX FS)
432 <1> ; ESI = Logical DOS Drive Description Table address
433 <1> ; EBX = Cluster (File R/W) Buffer address (max. 64KB)
434 <1> ; Only for SINGLIX FS:
435 <1> ; EDX = File Number (The 1st FDT address)
436 <1> ; OUTPUT ->
437 <1> ; cf = 1 -> Cluster can not be loaded at the buffer
438 <1> ; EAX > 0 -> Error number
439 <1> ; cf = 0 -> Cluster has been loaded at the buffer
440 <1> ;
441 <1> ; (Modified registers: EAX, ECX, EBX, EDX)
442 <1>
443 0000C16B 0FB64E13 <1> movzx ecx, byte [esi+LD_BPB+BPB_SecPerClust]
444 <1> ; CL = 1 = [esi+LD_FS_Reserved2] ; SectPerClust for Singlix FS
445 <1>
446 <1> read_file_sectors: ; 16/03/2016
447 <1>
448 <1> ; 19/12/2025
449 <1> %if 0
450 <1> ; cmp byte [esi+LD_FATType], 0
451 <1> ; 25/07/2022
452 <1> cmp [esi+LD_FATType], ch ; 0
453 <1> jna short read_fs_cluster
454 <1> %endif
455 <1>
456 <1> read_fat_file_sectors: ; 18/03/2016
457 0000C16F 83E802 <1> sub eax, 2 ; Beginning cluster number is always 2
458 0000C172 0FB65613 <1> movzx edx, byte [esi+LD_BPB+BPB_SecPerClust] ; 18/03/2016
459 0000C176 F7E2 <1> mul edx
460 0000C178 034668 <1> add eax, [esi+LD_DATABegin] ; absolute address of the cluster
461 <1>
462 <1> ; EAX = Disk sector address
463 <1> ; ECX = Sector count
464 <1> ; EBX = Buffer address
465 <1> ; (EDX = 0)
466 <1> ; ESI = Logical DOS drive description table address
467 <1>
468 0000C17B E863580000 <1> call disk_read
469 0000C180 7306 <1> jnc short rclust_retn
470 <1>
471 <1> ; 15/10/2016 (15h -> 17)
472 0000C182 B811000000 <1> mov eax, 17 ; Drive not ready or read error !
473 0000C187 C3 <1> retn
474 <1>
475 <1> rclust_retn:
476 0000C188 29C0 <1> sub eax, eax ; 0
477 0000C18A C3 <1> retn
478 <1>
479 <1> ; 19/12/2025
480 <1> %if 0
481 <1>
482 <1> read_fs_cluster:
483 <1> ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
484 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
485 <1> ; Singlix FS
486 <1>
487 <1> ; EAX = Cluster number is sector index number of the file (eax)
488 <1>
489 <1> ; EDX = File number is the first File Descriptor Table address
490 <1> ; of the file. (Absolute address of the FDT).
491 <1>
492 <1> ; eax = sector index (0 for the first sector)
493 <1> ; edx = FDT0 address
494 <1> ; 64 KB buffer = 128 sectors (limit)
495 <1> ; mov ecx, 128 ; maximum count of sectors (before eof)
496 <1> ; 25/07/2022
497 <1> sub ecx, ecx
498 <1> mov cl, 128
499 <1> ; call read_fs_sectors
500 <1> ; retn
501 <1> ; jmp short read_fs_sectors
502 <1>
503 <1> read_fs_sectors:
504 <1> ; 15/02/2016 (TRDOS 386 = TRDOS v2.0)
505 <1> stc
506 <1> retn
507 <1>
508 <1> %endif
509 <1>
510 <1> get_first_free_cluster:
511 <1> ; 21/12/2025 (TRDOS 386 kernel v2.0.10)
512 <1> ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
513 <1> ; 02/03/2016
514 <1> ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
515 <1> ; 26/10/2010 (DRV_FAT.ASM, 'proc_get_first_free_cluster')
516 <1> ; 10/07/2010
517 <1> ; INPUT ->
518 <1> ; ESI = Logical DOS Drive Description Table address
519 <1> ; OUTPUT ->
520 <1> ; cf = 1 -> Error code in AL (EAX)
521 <1> ; cf = 0 ->
522 <1> ; EAX = Cluster number
523 <1> ; If EAX = FFFFFFFFh -> no free space
524 <1> ; If the drive has FAT32 fs:
525 <1> ; EBX = FAT32 FSI sector buffer address (if > 0)
526 <1> ;
527 <1> ; (Modified registers: eax, ebx, ecx, edx)
528 <1> ;
529 <1>
530 0000C18B 8B4678 <1> mov eax, [esi+LD_Clusters]
531 0000C18E 40 <1> inc eax ; add eax, 1
532 0000C18F A314810100 <1> mov [gffc_last_free_cluster], eax
533 <1>
534 0000C194 31DB <1> xor ebx, ebx ; 0 ; 02/03/2016
535 <1>
536 0000C196 807E0302 <1> cmp byte [esi+LD_FATType], 2
537 0000C19A 760D <1> jna short loc_gffc_get_first_fat_free_cluster0
538 <1>
539 <1> ; 21/12/2025 - TRDOS 386 v2.0.10
540 <1> %if 1

```

```

541      <1>      ; FAT32 - FSINFO - first free cluster
542 0000C19C 31D2      <1>      xor     edx, edx
543 0000C19E 4A       <1>      dec     edx ; -1
544 0000C19F 87563E   <1>      xchg    edx, [esi+LD_BPB+BPB_Reserved+4]
545      <1>
546      <1> %else
547      <1> loc_gffc_get_first_fat32_free_cluster:
548      <1>      ; 02/03/2016
549      <1>      call   get_fat32_fsinfo_sector_parms
550      <1>      jc     short loc_gffc_get_first_fat_free_cluster0
551      <1> %endif
552      <1>
553      <1> loc_gffc_check_fsinfo_parms:
554      <1>      ; mov     ebx, DOSBootSectorBuff
555      <1>      ; cmp     dword [ebx], 41615252h
556      <1>      ; jne     short loc_gffc_fat32_fsinfo_err
557      <1>      ; cmp     dword [ebx+484], 61417272h
558      <1>      ; jne     short loc_gffc_fat32_fsinfo_err
559      <1>      ; mov     eax, [ebx+492] ; FSI_Next_Free
560      <1>      ; EAX = First free cluster
561      <1>      ; (from FAT32 FSInfo sector)
562 0000C1A2 89D0      <1>      mov     eax, edx ; FSI_Next_Free (First Free Cluster)
563 0000C1A4 83F8FF   <1>      cmp     eax, 0FFFFFFFh ; invalid (unknown) !
564 0000C1A7 7204      <1>      jb     short loc_gffc_get_first_fat_free_cluster1
565      <1>
566      <1>      ; Start from the 1st cluster of the FAT(32) file system
567      <1> loc_gffc_get_first_fat_free_cluster0:
568      <1>      ; mov     eax, 2
569      <1>      ; 25/07/2022
570 0000C1A9 29C0      <1>      sub     eax, eax
571 0000C1AB B002      <1>      mov     al, 2
572      <1>      ; xor     edx, edx
573      <1>
574      <1> loc_gffc_get_first_fat_free_cluster1:
575 0000C1AD 53          <1>      push    ebx ; 02/03/2016
576      <1>
577      <1> loc_gffc_get_first_fat_free_cluster2:
578 0000C1AE A3[10810100] <1>      mov     [gffc_first_free_cluster], eax
579 0000C1B3 A3[0C810100] <1>      mov     [gffc_next_free_cluster], eax
580      <1>
581      <1>      ; EBX = FAT32 FSINFO sector buffer address
582      <1>      ; (EBX = 0, if the drive has not got FAT32 fs or
583      <1>      ; FAT32 FSINFO sector buffer is invalid.)
584      <1>
585      <1> loc_gffc_get_first_fat_free_cluster3:
586 0000C1B8 E8BCFDFFFF <1>      call   get_next_cluster
587 0000C1BD 7307      <1>      jnc     short loc_gffc_get_first_fat_free_cluster4
588 0000C1BF 09C0      <1>      or      eax, eax
589 0000C1C1 740B      <1>      jz      short loc_gffc_first_free_fat_cluster_next
590 0000C1C3 5B          <1>      pop     ebx ; 02/03/2016
591 0000C1C4 F5          <1>      cmc     ; stc
592 0000C1C5 C3          <1>      retn
593      <1>
594      <1> loc_gffc_get_first_fat_free_cluster4:
595 0000C1C6 21C0      <1>      and     eax, eax ; next cluster value
596 0000C1C8 7504      <1>      jnz     short loc_gffc_first_free_fat_cluster_next
597 0000C1CA 89C8      <1>      mov     eax, ecx ; current (previous cluster) value
598 0000C1CC EB22      <1>      jmp     short loc_gffc_check_for_set
599      <1>
600      <1> loc_gffc_first_free_fat_cluster_next:
601 0000C1CE A1[0C810100] <1>      mov     eax, [gffc_next_free_cluster]
602 0000C1D3 3B05[14810100] <1>      cmp     eax, [gffc_last_free_cluster]
603 0000C1D9 7308      <1>      jnb     short retn_stc_from_get_first_free_cluster
604      <1> pass_gffc_last_cluster_eax_check:
605 0000C1DB 40          <1>      inc     eax ; add eax, 1
606 0000C1DC A3[0C810100] <1>      mov     [gffc_next_free_cluster], eax
607 0000C1E1 EBD5      <1>      jmp     short loc_gffc_get_first_fat_free_cluster3
608      <1>
609      <1> retn_stc_from_get_first_free_cluster:
610 0000C1E3 A1[10810100] <1>      mov     eax, [gffc_first_free_cluster]
611 0000C1E8 83F802   <1>      cmp     eax, 2
612 0000C1EB 7712      <1>      ja      short loc_gffc_check_previous_clusters
613 0000C1ED 29C0      <1>      sub     eax, eax
614 0000C1EF 48          <1>      dec     eax ; FFFFFFFFh
615      <1>
616      <1> loc_gffc_check_for_set:
617      <1>      ; 02/03/2016
618 0000C1F0 5B          <1>      pop     ebx
619      <1>
620      <1>      ; 21/12/2025
621      <1>      ;;;
622 0000C1F1 807E0302 <1>      cmp     byte [esi+LD_FATType], 2
623 0000C1F5 7603      <1>      jna     short loc_gffc_check_for_set_@
624      <1>
625 0000C1F7 89463E   <1>      mov     [esi+LD_BPB+BPB_Reserved+4], eax
626      <1>
627      <1> loc_gffc_check_for_set_@:
628      <1>      ;;;
629      <1>
630      <1>      ; EBX = FAT32 FSINFO sector buffer address
631      <1>      ; (EBX = 0, if the drive has not got FAT32 fs or
632      <1>      ; FAT32 FSINFO sector buffer is invalid.)
633      <1>
634 0000C1FA 09DB      <1>      or      ebx, ebx
635 0000C1FC 750D      <1>      jnz     short loc_gffc_set_ffree_fat32_cluster
636      <1>
637      <1>      ; cmp     byte [esi+LD_FATType], 3
638      <1>      ; jnb     short loc_gffc_set_ffree_fat32_cluster
639      <1>
640      <1>      ; xor     ebx, ebx ; 0
641      <1>
642      <1> loc_gffc_retn:
643 0000C1FE C3          <1>      retn
644      <1>
645      <1> loc_gffc_check_previous_clusters:
646 0000C1FF 48          <1>      dec     eax ; sub eax, 1
647 0000C200 A3[14810100] <1>      mov     [gffc_last_free_cluster], eax
648      <1>      ; mov     eax, 2
649      <1>      ; 25/07/2022
650 0000C205 31C0      <1>      xor     eax, eax
651 0000C207 B002      <1>      mov     al, 2
652      <1>      ; eax = 2
653      <1>      ; xor     edx, edx
654 0000C209 EBA3      <1>      jmp     short loc_gffc_get_first_fat_free_cluster2
655      <1>
656      <1> loc_gffc_set_ffree_fat32_cluster:
657      <1>      ; call   set_first_free_cluster
658      <1>      ; retn
659      <1>      ; jmp     short set_first_free_cluster
660      <1>
661      <1> set_first_free_cluster:
662      <1>      ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
663      <1>      ; 15/10/2016
664      <1>      ; 23/03/2016

```

```

665      <1>      ; 02/03/2016
666      <1>      ; 29/02/2016
667      <1>      ; 26/02/2016
668      <1>      ; 21/02/2016 (TRDOS 386 = TRDOS v2.0)
669      <1>      ; 21/08/2011 (DRV_FAT.ASM, 'proc_set_first_free_cluster')
670      <1>      ; 11/07/2010
671      <1>      ; INPUT ->
672      <1>      ; ESI = Logical DOS Drive Description Table address
673      <1>      ; EAX = First free cluster
674      <1>      ; EBX = FSINFO sector buffer address
675      <1>      ; ;If EBX > 0, it is FSINFO sector buffer address
676      <1>      ; ;EBX = 0, if FSINFO sector is not loaded
677      <1>      ; OUTPUT->
678      <1>      ; ESI = Logical DOS Drive Description Table address
679      <1>      ; If EBX > 0, it is FSINFO sector buffer address
680      <1>      ; EBX = 0, if FSINFO sector could not be loaded
681      <1>      ; CF = 1 -> Error code in AL (EAX)
682      <1>      ; CF = 0 -> first free cluster is successfully updated
683      <1>
684      <1>      ;cmp     byte [esi+LD_FATType], 3
685      <1>      ;jb      short loc_sffc_invalid_drive
686      <1>
687      <1>      ; Save First Free Cluster value for 'update_cluster'
688      0000C20B 89463E <1>      mov     [esi+LD_BPB+BPB_Reserved+4], eax ; First free cluster
689      <1>
690      <1>      ;or      ebx, ebx
691      <1>      ;jnz     short loc_sffc_read_fsinfo_sector
692      <1>
693      0000C20E 813B52526141 <1>      cmp     dword [ebx], 41615252h
694      0000C214 753C <1>      jne     short loc_sffc_read_fsinfo_sector
695      0000C216 81BBE4010000727241- <1>      cmp     dword [ebx+484], 61417272h
696      0000C21F 61 <1>
697      0000C220 7530 <1>      jne     short loc_sffc_read_fsinfo_sector
698      <1>
699      0000C222 3B83EC010000 <1>      cmp     eax, [ebx+492] ; FSI_Next_Free
700      0000C228 741E <1>      je      short loc_sffc_retn
701      <1>
702      <1>      loc_sffc_write_fsinfo_sector:
703      <1>      ; EBX = FSINFO sector buffer
704      <1>      ; [CFS_FAT32FSINFOSEC] is set in 'get_fat32_fsinfo_sector_parms'
705      0000C22A 8983EC010000 <1>      mov     [ebx+492], eax
706      0000C230 A1[24810100] <1>      mov     eax, [CFS_FAT32FSINFOSEC]
707      <1>      ;mov     ecx, 1
708      <1>      ; 25/07/2022
709      0000C235 31C9 <1>      xor     ecx, ecx
710      0000C237 FEC1 <1>      inc     cl
711      <1>      ; ecx = 1
712      0000C239 53 <1>      push    ebx
713      0000C23A E895570000 <1>      call    disk_write
714      0000C23F 7208 <1>      jc      short loc_sffc_read_fsinfo_sector_err1
715      0000C241 5B <1>      pop     ebx
716      <1>
717      0000C242 8B83EC010000 <1>      mov     eax, [ebx+492] ; First (Next) Free Cluster
718      <1>
719      0000C248 C3 <1>      loc_sffc_retn:
720      <1>      retn
721      <1>
722      <1>      ;loc_sffc_invalid_drive:
723      <1>      ; mov     eax, 0Fh ; MSDOS Error : Invalid drive
724      <1>      ; push    edx
725      <1>
726      <1>      loc_sffc_read_fsinfo_sector_err1:
727      <1>      ; 25/07/2022
728      <1>      ;mov     ebx, 0
729      <1>      ; 15/10/2016 (1Dh -> 18)
730      <1>      ; 23/03/2016 (1Dh)
731      0000C249 31C0 <1>      mov     eax, 18 ; Drive not ready or write error
732      0000C24B 89C3 <1>      xor     eax, eax
733      0000C24D B012 <1>      mov     ebx, eax ; 0
734      0000C24F F9 <1>      mov     al, 18
735      <1>      stc
736      0000C250 5A <1>      loc_sffc_read_fsinfo_sector_err2:
737      0000C251 C3 <1>      pop     edx
738      <1>      retn
739      <1>
740      <1>      loc_sffc_read_fsinfo_sector:
741      <1>      push    eax
742      0000C253 E84F050000 <1>      call    get_fat32_fsinfo_sector_parms
743      0000C258 72F6 <1>      jc      short loc_sffc_read_fsinfo_sector_err2
744      <1>
745      0000C25A 58 <1>      pop     eax
746      <1>      ; EDX = First (Next) Free Cluster value from FSINFO sector
747      <1>      ; EAX = First Free Cluster value from 'get_next_cluster'
748      <1>      ; (edx = old value)
749      0000C25B 39D0 <1>      cmp     eax, edx ; First free Cluster (eax = new value)
750      0000C25D 75CB <1>      jne     short loc_sffc_write_fsinfo_sector
751      <1>
752      0000C25F C3 <1>      retn
753      <1>
754      <1>      update_cluster:
755      <1>      ; 31/08/2024
756      <1>      ; 29/08/2024
757      <1>      ; 26/08/2024
758      <1>      ; 24/08/2024 (TRDOS 386 kernel v2.0.9)
759      <1>      ; 07/08/2022
760      <1>      ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
761      <1>      ; 23/10/2016
762      <1>      ; 23/03/2016
763      <1>      ; 02/03/2016
764      <1>      ; 01/03/2016
765      <1>      ; 29/02/2016
766      <1>      ; 27/02/2016
767      <1>      ; 26/02/2016
768      <1>      ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
769      <1>      ; 11/08/2011
770      <1>      ; 09/02/2005
771      <1>      ; INPUT ->
772      <1>      ; EAX = Cluster Number
773      <1>      ; ECX = New Cluster Value
774      <1>      ; ESI = Logical Dos Drive Parameters Table
775      <1>
776      <1>      ;
777      <1>      ; /// dword [FAT_ClusterCounter] ///
778      <1>
779      <1>      ; OUTPUT ->
780      <1>      ; cf = 0 -> No Error, EAX is valid
781      <1>      ; cf = 1 & EAX = 0 -> End Of Cluster Chain
782      <1>      ; cf = 1 & EAX > 0 -> Error
783      <1>      ; (ECX -> any value)
784      <1>      ; EAX = Next Cluster
785      <1>      ; ECX = New Cluster Value
786      <1>
787      <1>      ;
788      <1>      ; /// [FAT_ClusterCounter] is updated,
789      <1>      ; /// decreased when a free cluster is assigned,

```

```

788      ;          /// increased if an assigned cluster is freed.
789      ;
790      ;
791      ; (Modified registers: EAX, EBX, -ECX-, EDX)
792      ;
793 0000C260 A3[727E0100]      mov     [FAT_CurrentCluster], eax
794 0000C265 890D[18810100]    mov     [ClusterValue], ecx
795      ;
796      loc_update_cluster_check_fat_buffer:
797 0000C26B 8A1E            mov     bl, [esi+LD_Name]
798 0000C26D 381D[777E0100]    cmp     [FAT_BuffDrvName], bl
799 0000C273 7418            je      short loc_update_cluster_check_fat_type
800 0000C275 803D[767E0100]02      cmp     byte [FAT_BuffValidData], 2
801      ;je      loc_uc_save_fat_buffer
802      ; 25/07/2022
803 0000C27C 7502            jne     short loc_uc_reset_fat_buffer_validation
804 0000C27E EB66            jmp     loc_uc_save_fat_buffer
805      ;
806      loc_uc_reset_fat_buffer_validation:
807 0000C280 C605[767E0100]00  mov     byte [FAT_BuffValidData], 0
808      ;
809      loc_uc_check_fat_type_reset_drvname:
810 0000C287 881D[777E0100]    mov     [FAT_BuffDrvName], bl
811      ;
812      loc_update_cluster_check_fat_type:
813 0000C28D 29D2            sub     edx, edx ; 26/02/2016
814 0000C28F 8A5E03          mov     bl, [esi+LD_FATType]
815 0000C292 83F802          cmp     eax, 2
816 0000C295 7218            jb      short update_cluster_inv_data
817      ;
818      ;;;
819      ; 24/08/2024
820      ; edx = 0 ; 24/08/2024
821 0000C297 8B4E78          mov     ecx, [esi+LD_Clusters]
822 0000C29A 41            inc     ecx
823 0000C29B 890D[827E0100]    mov     [LastCluster], ecx
824      ;
825 0000C2A1 39C8            cmp     eax, ecx
826      ;ja      short return_uc_fat32_stc ; 25/07/2022
827      ; edx = 0 ; (must be -1 or > 0 after here)
828 0000C2A3 7717            ja      short return_uc_fat_stc
829      ;;;
830      ;
831 0000C2A5 80FB02          cmp     bl, 2
832      ;ja      update_fat32_cluster
833      ; 25/07/2022
834 0000C2A8 7608            jna     short loc_uc_check_fat_type_1
835 0000C2AA E97B010000      jmp     update_fat32_cluster
836      ;
837      ; 24/08/2024
838      ;loc_uc_check_fat_type_1:
839      ; ;cmp     bl, 1
840      ; ;jb      short update_cluster_inv_data
841      ; ;mov     ecx, [esi+LD_Clusters]
842      ; ;inc     ecx
843      ; ;mov     [LastCluster], ecx
844      ; ;cmp     eax, ecx ; dword [LastCluster]
845      ; ;ja      return_uc_fat_stc
846      ; ; 25/07/2022
847      ; ;jna     short loc_uc_check_fat_type_2
848      ; ; 24/08/2024
849      ; ; edx = 0 ; (must be -1 or > 0 after here)
850      ; ; jmp     return_uc_fat_stc
851      ;
852      ; 25/07/2022
853      update_cluster_inv_data:
854      ; ;mov     eax, 0Dh
855      ; ;mov     al, 0Dh ; Invalid Data
856      ; ; 31/08/2024
857 0000C2AF B01D          mov     al, ERR_INV_DATA ; 29 ; Invalid Data
858 0000C2B1 C3            retn
859      ;
860      ; 24/08/2024
861      loc_uc_check_fat_type_1:
862      loc_uc_check_fat_type_2:
863      ; TRDOS v1 has a FAtal bug here !
864      ; ; or bl, bl ; cmp bl, 0
865      ; ; jz short update_fat12_cluster
866      ; !! It would destroy FAT12 floppy disk fs here !!
867      ; ('A:' disks of TRDOS v1 operating system project
868      ; had 'singlix fs', so, I could not differ this mistake
869      ; on a drive 'A:')
870 0000C2B2 80FB01          cmp     bl, 1 ; correct comparison is this !
871      ; ;jna     update_fat12_cluster
872      ; ; 25/07/2022
873 0000C2B5 770B            ja      short update_fat16_cluster
874 0000C2B7 E9CF000000      jmp     update_fat12_cluster
875      ;
876      ; 24/08/2024
877      ;return_uc_fat16_stc:
878      ; 25/07/2022
879      return_uc_fat_stc:
880      ; 24/08/2024
881      ; edx = 0
882 0000C2BC 4A            dec     edx
883      ; edx = -1 ; 0FFFFFFFh
884      ; ecx > 0
885      return_uc_fat16_stc: ; 24/08/2024
886      loc_fat_buffer_stc_0: ; 24/08/2024
887      ; 01/03/2016
888 0000C2BD 31C0          xor     eax, eax
889 0000C2BF F9            stc
890 0000C2C0 EB70            jmp     short loc_fat_buffer_stc_1
891      ;
892      update_fat16_cluster:
893      pass_uc_fat16_errc:
894      ;sub     edx, edx
895      ; edx = 0
896 0000C2C2 BB00030000      mov     ebx, 300h ; 768
897 0000C2C7 F7F3            div     ebx
898      ; EAX = Count of 3 FAT sectors
899      ; DX = Cluster index in FAT buffer
900      ;mov     bx, dx
901      ; 25/07/2022
902 0000C2C9 89D3          mov     ebx, edx
903      ;shl     bx, 1 ; Multiply by 2
904      ; 25/07/2022
905 0000C2CB D1E3          shl     ebx, 1
906 0000C2CD 66BA0300      mov     dx, 3
907 0000C2D1 F7E2            mul     edx
908      ; EAX = FAT Sector
909      ; EDX = 0
910      ; EBX = Byte offset in FAT buffer
911 0000C2D3 8A0D[767E0100]    mov     cl, [FAT_BuffValidData]

```

```

912 0000C2D9 80F902    <1>    cmp     cl, 2
913 0000C2DC 7518      <1>    jne     short loc_uc_check_fat16_buff_sector_load
914                                <1>
915                                <1> loc_uc_check_fat16_buff_sector_save:
916 0000C2DE 3B05[7A7E0100] <1>    cmp     eax, [FAT_BuffSector]
917                                <1>    ;jne     short loc_uc_save_fat_buffer
918                                <1>    ;jmp     short loc_update_fat16_cell
919                                <1>    ; 07/08/2022
920 0000C2E4 741D      <1>    je      short loc_update_fat16_cell
921                                <1>    ;jmp     loc_uc_save_fat_buffer
922                                <1>
923                                <1>    ; 07/08/2022
924                                <1> loc_uc_save_fat_buffer:
925                                <1>    ; byte [FAT_BuffValidData] = 2
926 0000C2E6 E8E9010000 <1>    call    save_fat_buffer
927 0000C2EB 7267      <1>    jc      short loc_fat_sectors_rw_error2
928                                <1>    ;mov     byte [FAT_BuffValidData], 1
929 0000C2ED A1[727E0100] <1>    mov     eax, [FAT_CurrentCluster]
930                                <1>    ;mov     ecx, [ClusterValue]
931                                <1>    ;jmp     short loc_update_cluster_check_fat_buffer
932 0000C2F2 8A1E      <1>    mov     bl, [esi+LD_Name] ; 01/03/2016
933 0000C2F4 EB8A      <1>    jmp     loc_uc_reset_fat_buffer_validation
934                                <1>
935                                <1> loc_uc_check_fat16_buff_sector_load:
936 0000C2F6 80F901 <1>    cmp     cl, 1 ; byte [FAT_BuffValidData]
937 0000C2F9 7560      <1>    jne     short loc_uc_load_fat_sectors
938 0000C2FB 3B05[7A7E0100] <1>    cmp     eax, [FAT_BuffSector]
939 0000C301 7558      <1>    jne     short loc_uc_load_fat_sectors
940                                <1>
941                                <1> loc_update_fat16_cell:
942                                <1> loc_update_fat16_buffer:
943 0000C303 81C3001C0900 <1>    add     ebx, FAT_Buffer ; 26/02/2016
944                                <1>    ;movzx   eax, word [ebx]
945 0000C309 668B03 <1>    mov     ax, [ebx]
946                                <1>    ; 01/03/2016
947 0000C30C 89C2      <1>    mov     edx, eax ; old value of the cluster
948 0000C30E A3[727E0100] <1>    mov     [FAT_CurrentCluster], eax
949 0000C313 8B0D[18810100] <1>    mov     ecx, [ClusterValue] ; 32 bits
950 0000C319 66890B <1>    mov     [ebx], cx ; 16 bits !
951                                <1>
952 0000C31C C605[767E0100]02 <1>    mov     byte [FAT_BuffValidData], 2
953                                <1>
954 0000C323 6683F802 <1>    cmp     ax, 2
955 0000C327 7294      <1>    jb      short return_uc_fat16_stc
956 0000C329 3B05[827E0100] <1>    cmp     eax, [LastCluster]
957 0000C32F 778C      <1>    ja      short return_uc_fat16_stc
958                                <1>
959                                <1> loc_fat_buffer_updated:
960                                <1>    ; 01/03/2016
961 0000C331 F8      <1>    clc
962                                <1> loc_fat_buffer_stc_1:
963 0000C332 9C      <1>    pushf
964 0000C333 21C9 <1>    and     ecx, ecx
965 0000C335 7506 <1>    jnz     short loc_fat_buffer_updated_1
966                                <1>
967                                <1>    ; 01/03/2016
968                                <1>    ; new value of the cluster = 0 (free)
969                                <1>    ; increase free(d) cluster count
970 0000C337 FF05[7E7E0100] <1>    inc     dword [FAT_ClusterCounter]
971                                <1>
972                                <1> loc_fat_buffer_updated_1: ; new value of the cluster > 0
973 0000C33D 09D2 <1>    or      edx, edx ; 02/03/2016
974 0000C33F 7506 <1>    jnz     short loc_fat_buffer_updated_2
975                                <1>    ; old value of the cluster = 0 (it was free cluster)
976                                <1>    ; decrease free(d) cluster count
977 0000C341 FF0D[7E7E0100] <1>    dec     dword [FAT_ClusterCounter] ; it may be negative number
978                                <1>
979                                <1> loc_fat_buffer_updated_2:
980 0000C347 9D      <1>    popf
981 0000C348 C3      <1>    retn
982                                <1>
983                                <1>    ; 25/07/2022
984                                <1> loc_fat_sectors_rw_error1:
985                                <1>    ;mov     byte [FAT_BuffValidData], 0
986                                <1>    ; 23/10/2016 (15h -> 17)
987                                <1>    ; 23/03/2016
988 0000C349 B811000000 <1>    mov     eax, 17 ; Drive not ready or read error
989 0000C34E 8825[767E0100] <1>    mov     [FAT_BuffValidData], ah ; 0
990                                <1>
991                                <1> loc_fat_sectors_rw_error2:
992                                <1>    ;mov     eax, error code
993                                <1>    ;mov     edx, 0
994 0000C354 8B0D[18810100] <1>    mov     ecx, [ClusterValue]
995 0000C35A C3      <1>    retn
996                                <1>
997                                <1>    ; 25/07/2022
998                                <1> loc_uc_load_fat_sectors:
999 0000C35B A3[7A7E0100] <1>    mov     [FAT_BuffSector], eax
1000                                <1>
1001                                <1> load_uc_fat_sectors_zero:
1002 0000C360 034660 <1>    add     eax, [esi+LD_FATBegin]
1003 0000C363 BB001C0900 <1>    mov     ebx, FAT_Buffer
1004 0000C368 B903000000 <1>    mov     ecx, 3
1005 0000C36D E871560000 <1>    call    disk_read
1006 0000C372 72D5 <1>    jc      short loc_fat_sectors_rw_error1
1007                                <1>
1008 0000C374 C605[767E0100]01 <1>    mov     byte [FAT_BuffValidData], 1
1009 0000C37B A1[727E0100] <1>    mov     eax, [FAT_CurrentCluster]
1010 0000C380 8B0D[18810100] <1>    mov     ecx, [ClusterValue]
1011 0000C386 E902FFFFFF <1>    jmp     loc_update_cluster_check_fat_type
1012                                <1>
1013                                <1> update_fat12_cluster:
1014                                <1> pass_uc_fat12_errc:
1015                                <1>    ;sub     edx, edx
1016 0000C38B BB00040000 <1>    mov     ebx, 400h ; 1024
1017 0000C390 F7F3 <1>    div     ebx
1018                                <1>    ; EAX = Count of 3 FAT sectors
1019                                <1>    ; DX = Cluster index in FAT buffer
1020                                <1>    ;mov     cx, 3
1021                                <1>    ; 25/07/2022
1022 0000C392 29C9 <1>    sub     ecx, ecx
1023 0000C394 B103 <1>    mov     cl, 3
1024                                <1>    ; ecx = 3
1025                                <1>    ;mov     bx, ax
1026 0000C396 89C3 <1>    mov     ebx, eax
1027                                <1>    ;mov     ax, cx ; 3
1028 0000C398 89C8 <1>    mov     eax, ecx
1029                                <1>    ;mul     dx ; Multiply by 3
1030 0000C39A F7E2 <1>    mul     edx
1031                                <1>    ;shr     ax, 1 ; Divide by 2
1032 0000C39C D1E8 <1>    shr     eax, 1
1033                                <1>    ;xchg     bx, ax
1034 0000C39E 93 <1>    xchg     ebx, eax
1035                                <1>    ; EAX = Count of 3 FAT sectors

```

```

1036 <1> ; EBX = Byte Offset in FAT buffer
1037 <1> ; mul cx ; 3 * AX
1038 0000C39F F7E1 <1> mul ecx ; 3 * EAX
1039 <1> ; EAX = FAT Beginning Sector
1040 <1> ; EDX = 0
1041 0000C3A1 8A0D[767E0100] <1> mov cl, [FAT_BuffValidData]
1042 <1> ; TRDOS v1 has a FATA! bug here !
1043 <1> ; (it does not have 'cmp cl, 2' instruction here !
1044 <1> ; while 'jne' is existing !)
1045 0000C3A7 80F902 <1> cmp cl, 2 ; 2 = dirty buffer (must be written to disk)
1046 0000C3AA 750D <1> jne short loc_uc_check_fat12_buff_sector_load
1047 <1>
1048 <1> loc_uc_check_fat12_buff_sector_save:
1049 0000C3AC 3B05[7A7E0100] <1> cmp eax, [FAT_BuffSector]
1050 <1> ; jne short loc_uc_save_fat_buffer
1051 <1> ; jmp short loc_update_fat12_cell
1052 <1> ; 07/08/2022
1053 0000C3B2 7412 <1> je short loc_update_fat12_cell
1054 0000C3B4 E92DFFFFFF <1> jmp loc_uc_save_fat_buffer
1055 <1>
1056 <1> loc_uc_check_fat12_buff_sector_load:
1057 0000C3B9 80F901 <1> cmp cl, 1 ; byte ptr [FAT_BuffValidData]
1058 0000C3BC 759D <1> jne short loc_uc_load_fat_sectors
1059 0000C3BE 3B05[7A7E0100] <1> cmp eax, [FAT_BuffSector]
1060 0000C3C4 7595 <1> jne short loc_uc_load_fat_sectors
1061 <1> ; 07/08/2022
1062 <1> ; je short loc_update_fat12_cell
1063 <1> ; jmp loc_uc_load_fat_sectors
1064 <1>
1065 <1> loc_update_fat12_cell:
1066 0000C3C6 81C3001C0900 <1> add ebx, FAT_Buffer ; 26/02/2016
1067 <1> ; mov cx, [FAT_CurrentCluster]
1068 <1> ; 25/07/2022
1069 0000C3CC 8B0D[727E0100] <1> mov ecx, [FAT_CurrentCluster]
1070 <1> ; shr cx, 1
1071 <1> ; 25/07/2022
1072 0000C3D2 D1E9 <1> shr ecx, 1
1073 0000C3D4 668B03 <1> mov ax, [ebx]
1074 <1> ; mov dx, ax
1075 0000C3D7 89C2 <1> mov edx, eax ; 25/07/2022
1076 <1> ; 24/08/2024 (*)
1077 0000C3D9 8B0D[18810100] <1> mov ecx, [ClusterValue] ; 32 bits
1078 0000C3DF 7336 <1> jnc short uc_fat12_nc_even
1079 <1>
1080 <1> ; 26/08/2024
1081 <1> ; ODD cluster number
1082 <1> ; eax = current value (before updated)
1083 <1> ; ecx = new value of the cluster
1084 <1> ; of the cluster
1085 <1> ; low 4 bit of al is high 4 bit
1086 <1> ; of the previous cluster
1087 <1> ; (it must not be overwritten)
1088 <1>
1089 <1> ; and ax, 0Fh
1090 <1> ; 25/07/2022
1091 0000C3E1 240F <1> and al, 0Fh
1092 <1> ; 24/08/2024 (*)
1093 <1> ; mov ecx, [ClusterValue] ; 32 bits
1094 <1> ; shl cx, 4
1095 0000C3E3 C1E104 <1> shl ecx, 4
1096 <1> ; or cx, ax
1097 0000C3E6 08C1 <1> or cl, al ; 25/07/2022
1098 <1> ; mov ax, dx
1099 0000C3E8 89D0 <1> mov eax, edx
1100 0000C3EA 66890B <1> mov [ebx], cx ; 16 bits !
1101 <1> ; shr ax, 4 ; al(bit4..7)+ah(bit0..7)
1102 <1> ; 25/07/2022
1103 0000C3ED C1E804 <1> shr eax, 4
1104 <1>
1105 <1> update_fat12_buffer:
1106 <1> ;;;
1107 <1> ; 24/08/2024
1108 0000C3F0 8B0D[18810100] <1> mov ecx, [ClusterValue]
1109 <1> ;;;
1110 0000C3F6 A3[727E0100] <1> mov [FAT_CurrentCluster], eax
1111 0000C3FB 89C2 <1> mov edx, eax ; 01/03/2016
1112 0000C3FD C605[767E0100]02 <1> mov byte [FAT_BuffValidData], 2
1113 0000C404 6683F802 <1> cmp ax, 2
1114 0000C408 724A <1> jb short return_uc_fat12_stc
1115 0000C40A 3B05[827E0100] <1> cmp eax, [LastCluster]
1116 0000C410 7742 <1> ja short return_uc_fat12_stc
1117 0000C412 E91AFFFFFF <1> jmp loc_fat_buffer_updated
1118 <1>
1119 <1> uc_fat12_nc_even:
1120 <1> ; 26/08/2024
1121 <1> ; EVEN cluster number
1122 <1> ; eax = current value (before updated)
1123 <1> ; ecx = new value of the cluster
1124 <1> ; of the cluster
1125 <1> ; high 4 bit of ah is low 4 bit
1126 <1> ; of the next cluster (it must not be overwritten)
1127 <1>
1128 0000C417 662500F0 <1> and ax, 0F000h
1129 <1> ; 24/08/2024 (*)
1130 <1> ; mov ecx, [ClusterValue] ; 32 bits
1131 0000C41B 80E50F <1> and ch, 0Fh
1132 <1> ; or cx, ax
1133 <1> ; 25/07/2022
1134 0000C41E 09C1 <1> or ecx, eax
1135 <1> ; mov ax, dx
1136 0000C420 89D0 <1> mov eax, edx
1137 0000C422 66890B <1> mov [ebx], cx ; 16 bits !
1138 0000C425 80E40F <1> and ah, 0Fh ; al(bit0..7)+ah(bit0..3)
1139 0000C428 EBC6 <1> jmp short update_fat12_buffer
1140 <1>
1141 <1> update_fat32_cluster:
1142 <1> ; edx = 0 ; 24/08/2024
1143 <1> ; mov ecx, [esi+LD_clusters]
1144 <1> ; inc ecx
1145 <1> ; mov [LastCluster], ecx
1146 <1> ; 24/08/2024
1147 <1> ; ecx = [LastCluster]
1148 <1>
1149 <1> ; 24/08/2024
1150 <1> ; cmp eax, ecx
1151 <1> ; ja short return_uc_fat32_stc ; 25/07/2022
1152 <1> ; 24/08/2024
1153 <1> ; edx = 0 ; (must be -1 or > 0 after here)
1154 <1> ; ja short return_uc_fat_stc
1155 <1>
1156 <1> pass_uc_fat32_errc:
1157 <1> ; sub edx, edx
1158 0000C42A BB80010000 <1> mov ebx, 180h ; 384
1159 0000C42F F7F3 <1> div ebx

```

```

1160      <1>      ; EAX = Count of 3 FAT sectors
1161      <1>      ; DX = Cluster index in FAT buffer
1162      0000C431 89D3      <1>      mov     ebx, edx
1163      0000C433 C1E302    <1>      shl     ebx, 2 ; Multiply by 4
1164      <1>      ;mov     edx, 3
1165      <1>      ; 25/07/2022
1166      <1>      ;xor     dh, dh
1167      <1>      ;mov     dl, 3
1168      0000C436 66BA0300  <1>      mov     dx, 3
1169      0000C43A F7E2      <1>      mul     edx
1170      <1>      ; EBX = Cluster Offset in FAT buffer
1171      <1>      ; EAX = FAT Sector
1172      <1>      ; EDX = 0
1173      0000C43C 8A0D[767E0100] <1>      mov     cl, [FAT_BuffValidData]
1174      0000C442 80F902    <1>      cmp     cl, 2
1175      0000C445 7512      <1>      jne     short loc_uc_check_fat32_buff_sector_load
1176      <1>
1177      <1>      loc_uc_check_fat32_buff_sector_save:
1178      0000C447 3B05[7A7E0100] <1>      cmp     eax, [FAT_BuffSector]
1179      <1>      ;jne     loc_uc_save_fat_buffer
1180      <1>      ;jmp     short loc_update_fat32_cell
1181      <1>      ; 25/07/2022
1182      0000C44D 741C      <1>      je      short loc_update_fat32_cell
1183      0000C44F E992FEFFFF <1>      jmp     loc_uc_save_fat_buffer
1184      <1>
1185      <1>      return_uc_fat12_stc:
1186      <1>      return_uc_fat32_stc:
1187      <1>      ; 24/08/2024
1188      0000C454 E964FEFFFF <1>      jmp     loc_fat_buffer_stc_0 ; (*)
1189      <1>      ; 25/07/2022
1190      <1>      ;sub     eax, eax
1191      <1>      ;stc
1192      <1>      ;jmp     loc_fat_buffer_stc_1 ; (*)
1193      <1>
1194      <1>      loc_uc_check_fat32_buff_sector_load:
1195      0000C459 80F901    <1>      cmp     cl, 1 ; byte [FAT_BuffValidData]
1196      <1>      ;jne     loc_uc_load_fat_sectors
1197      <1>      ; 25/07/2022
1198      0000C45C 7508      <1>      jne     short loc_uc_load_fat_sects
1199      0000C45E 3B05[7A7E0100] <1>      cmp     eax, [FAT_BuffSector]
1200      <1>      ;jne     loc_uc_load_fat_sectors
1201      <1>      ; 25/07/2022
1202      0000C464 7405      <1>      je      short loc_update_fat32_cell
1203      <1>      loc_uc_load_fat_sects:
1204      0000C466 E9F0FEFFFF <1>      jmp     loc_uc_load_fat_sectors
1205      <1>
1206      <1>      loc_update_fat32_cell:
1207      <1>      loc_update_fat32_buffer:
1208      0000C46B 81C3001C0900 <1>      add     ebx, FAT_Buffer ; 26/02/2016
1209      0000C471 8B03      <1>      mov     eax, [ebx]
1210      0000C473 25FFFFFF0F <1>      and     eax, 0FFFFFFFh ; 28 bit cluster value
1211      <1>
1212      0000C478 8B15[727E0100] <1>      mov     edx, [FAT_CurrentCluster] ; 01/03/2016
1213      <1>
1214      0000C47E A3[727E0100]    <1>      mov     [FAT_CurrentCluster], eax
1215      0000C483 8B0D[18810100] <1>      mov     ecx, [ClusterValue]
1216      <1>      ;;;
1217      <1>      ; 29/08/2024
1218      0000C489 81E1FFFFFF0F <1>      and     ecx, 0FFFFFFFh ; 28 bit cluster value
1219      <1>      ;;;
1220      0000C48F 890B      <1>      mov     [ebx], ecx ; 29/02/2016
1221      <1>
1222      0000C491 C605[767E0100]02 <1>      mov     byte [FAT_BuffValidData], 2
1223      <1>
1224      <1>      ; 01/03/2016
1225      0000C498 21C0      <1>      and     eax, eax ; was it free cluster ?
1226      0000C49A 7513      <1>      jnz     short loc_upd_fat32_c0
1227      <1>
1228      <1>      ;or      ecx, ecx ; it will be left free ?!
1229      <1>      ;jz      short loc_upd_fat32_c3
1230      <1>
1231      0000C49C 3B563E    <1>      cmp     edx, [esi+LD_BPB+BPB_Reserved+4] ; First free cluster
1232      0000C49F 751F      <1>      jne     short loc_upd_fat32_c3
1233      <1>
1234      0000C4A1 3B15[827E0100] <1>      cmp     edx, [LastCluster]
1235      0000C4A7 7206      <1>      jnb     short loc_upd_fat32_c0
1236      <1>
1237      <1>      ;mov     edx, 2 ; rewind !
1238      <1>      ; 25/07/2022
1239      0000C4A9 29D2      <1>      sub     edx, edx
1240      0000C4AB B202      <1>      mov     dl, 2
1241      0000C4AD EB0E      <1>      jmp     short loc_upd_fat32_c2
1242      <1>
1243      <1>      loc_upd_fat32_c0:
1244      0000C4AF FF463E    <1>      inc     dword [esi+LD_BPB+BPB_Reserved+4] ; set it to next cluster
1245      0000C4B2 EB0C      <1>      jmp     short loc_upd_fat32_c3
1246      <1>
1247      <1>      loc_upd_fat32_c1:
1248      0000C4B4 09C9      <1>      or      ecx, ecx ; will it be free cluster ?
1249      0000C4B6 7508      <1>      jnz     short loc_upd_fat32_c3
1250      <1>
1251      0000C4B8 3B563E    <1>      cmp     edx, [esi+LD_BPB+BPB_Reserved+4] ; First free cluster
1252      0000C4BB 7303      <1>      jnb     short loc_upd_fat32_c3
1253      <1>
1254      <1>      loc_upd_fat32_c2:
1255      0000C4BD 89563E    <1>      mov     [esi+LD_BPB+BPB_Reserved+4], edx
1256      <1>
1257      <1>      loc_upd_fat32_c3:
1258      0000C4C0 89C2      <1>      mov     edx, eax
1259      <1>
1260      <1>      loc_upd_fat32_c4:
1261      0000C4C2 83F802    <1>      cmp     eax, 2
1262      0000C4C5 728D      <1>      jnb     short return_uc_fat32_stc ; 25/07/2022
1263      <1>
1264      <1>      pass_uc_fat32_c_zero_check_2:
1265      0000C4C7 3B05[827E0100] <1>      cmp     eax, [LastCluster]
1266      0000C4CD 7785      <1>      ja      short return_uc_fat32_stc ; 25/07/2022
1267      <1>
1268      0000C4CF E95DFEFFFF <1>      jmp     loc_fat_buffer_updated
1269      <1>
1270      <1>      save_fat_buffer:
1271      <1>      ; 19/12/2025 (TRDOS 386 v2.0.10)
1272      <1>      ; 31/08/2024 (TRDOS 386 v2.0.9)
1273      <1>      ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
1274      <1>      ; 15/10/2016
1275      <1>      ; 01/03/2016
1276      <1>      ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
1277      <1>      ; 11/08/2011
1278      <1>      ; 09/02/2005
1279      <1>      ; INPUT ->
1280      <1>      ; None
1281      <1>      ; OUTPUT ->
1282      <1>      ; cf = 0 -> OK.
1283      <1>      ; cf = 1 -> error code in AL (EAX)

```

```

1284      <1>      ;
1285      <1>      ; EBX = FAT_Buffer address
1286      <1>      ;
1287      <1>      ; (EAX, EDX, ECX, EBX will be modified)
1288      <1>
1289      <1>      ;cmp    byte [FAT_BuffValidData], 2
1290      <1>      ;je     short loc_save_fat_buff
1291      <1>
1292      <1> ;loc_save_fat_buffer_retn:
1293      <1> ; xor     eax, eax
1294      <1> ; retn
1295      <1>
1296      <1> loc_save_fat_buff:
1297      <1> xor     edx, edx
1298      <1> mov     dh, [FAT_BuffDrvName]
1299      <1> cmp     dh, 'A'
1300      <1> jb     short loc_save_fat_buffer_inv_data_retn
1301      <1> sub     dh, 'A'
1302      <1> push    esi ; *
1303      <1> mov     esi, Logical_DOSDisks
1304      <1> add     esi, edx
1305      <1>
1306      <1> mov     dl, [esi+LD_FATType]
1307      <1> ; 19/12/2025
1308      <1> %if 0
1309      <1> and     dl, dl
1310      <1> jz     short loc_save_fat_buffer_inv_data_pop_retn
1311      <1> %endif
1312      <1> mov     eax, [FAT_BuffSector]
1313      <1> cmp     dl, 2
1314      <1> ja     short loc_save_fat32_buff
1315      <1>
1316      <1> loc_save_fat_12_16_buff:
1317      <1> ; 01/03/2016
1318      <1> ; TRDOS v1 has a FATA! bug here!
1319      <1> ; correct code: mov dx, word ptr [FAT_BuffSector]+2
1320      <1> ; (DX:AX in TRDOS v1 -> EAX in TRDOS v2)
1321      <1> ;
1322      <1> movzx   ecx, word [esi+LD_BPB+FATSecs]
1323      <1> sub     ecx, eax
1324      <1> ; TRDOS v1 has a bug here... ('pop esi' was forgotten!)
1325      <1> ;jna     short loc_save_fat_buffer_inv_data_retn ; wrong addr!
1326      <1> jna     short loc_save_fat_buffer_inv_data_pop_retn ; correct addr.
1327      <1> ; 25/07/2022
1328      <1> ;jmp     short loc_save_fat_buffer_check_rs3
1329      <1>
1330      <1> loc_save_fat_buffer_check_rs3:
1331      <1> ; 25/07/2022
1332      <1> sub     ebx, ebx
1333      <1> mov     bl, 3
1334      <1> ;cmp     ecx, 3
1335      <1> cmp     ecx, ebx ; 3
1336      <1> jna     short loc_save_fat_buff_continue
1337      <1> mov     ecx, ebx ; mov ecx, 3
1338      <1> loc_save_fat_buff_continue:
1339      <1> mov     ebx, FAT_Buffer
1340      <1> add     eax, [esi+LD_FATBegin]
1341      <1> push    ecx
1342      <1> call    disk_write
1343      <1> pop     ecx
1344      <1> jc     short loc_save_FAT_buff_write_err
1345      <1>
1346      <1> cmp     byte [esi+LD_FATType], 2
1347      <1> jna     short loc_calc_2nd_fat12_16_addr
1348      <1>
1349      <1> loc_calc_2nd_fat32_addr:
1350      <1> mov     eax, [esi+LD_BPB+FAT32_FAT_Size]
1351      <1> jmp     short loc_calc_2nd_fat_addr
1352      <1>
1353      <1> ; 25/07/2022
1354      <1> loc_save_fat32_buff:
1355      <1> mov     ecx, [esi+LD_BPB+FAT32_FAT_Size]
1356      <1> sub     ecx, eax
1357      <1> ja     short loc_save_fat_buffer_check_rs3
1358      <1>
1359      <1> loc_save_fat_buffer_inv_data_pop_retn:
1360      <1> pop     esi ; *
1361      <1> loc_save_fat_buffer_inv_data_retn:
1362      <1> ;mov     eax, 0dh ; Invalid DATA
1363      <1> ; 25/07/2022
1364      <1> sub     eax, eax
1365      <1> ;mov     al, 0dh ; Invalid Data
1366      <1> ; 31/08/2024
1367      <1> mov     al, ERR_INV_DATA ; 29 ; Invalid Data
1368      <1> stc     ; cf = 1
1369      <1> retn
1370      <1>
1371      <1> loc_calc_2nd_fat12_16_addr:
1372      <1> movzx   eax, word [esi+LD_BPB+FATSecs]
1373      <1>
1374      <1> loc_calc_2nd_fat_addr:
1375      <1> add     eax, [esi+LD_FATBegin]
1376      <1> add     eax, [FAT_BuffSector]
1377      <1> mov     ebx, FAT_Buffer
1378      <1> ; ecx = 1 to 3
1379      <1> call    disk_write
1380      <1> jc     short loc_save_FAT_buff_write_err
1381      <1> ; valid buffer (1 = valid but do not save)
1382      <1> mov     byte [FAT_BuffValidData], 1
1383      <1>
1384      <1> loc_save_FAT_buff_write_err:
1385      <1> pop     esi ; *
1386      <1> mov     ebx, FAT_Buffer
1387      <1> ; 15/10/2016 (1Dh -> 18)
1388      <1> ; 23/03/2016 (1Dh)
1389      <1> mov     eax, 18 ; Drive not ready or write error
1390      <1> retn
1391      <1>
1392      <1> calculate_fat_freespace:
1393      <1> ; 21/12/2025 (TRDOS 386 kernel v2.0.10)
1394      <1> ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
1395      <1> ; 23/03/2016
1396      <1> ; 02/03/2016
1397      <1> ; 01/03/2016
1398      <1> ; 29/02/2016
1399      <1> ; 22/02/2016 (TRDOS 386 = TRDOS v2.0)
1400      <1> ; 30/04/2011
1401      <1> ; 03/04/2010
1402      <1> ; 2005
1403      <1> ; INPUT ->
1404      <1> ; EAX = Cluster count to be added or subtracted
1405      <1> ; If BH = FFh, ESI = TR-DOS Logical Drive Description Table
1406      <1> ; If BH < FFh, BH = TR-DOS Logical Drive Number
1407      <1> ; BL:

```



```

1408      ;      0 = Calculate, 1 = Add, 2 = Subtract, 3 = Get (Not Set/calc)
1409      ;      OUTPUT ->
1410      ;      EAX = Free Space in sectors
1411      ;      ESI = Logical Dos Drive Description Table address
1412      ;      BH = Logical Dos Drive Number (same with input value of BH)
1413      ;      BL = Type of operation (same with input value of BL)
1414      ;      ECX = 0 -> valid
1415      ;      ECX > 0 -> error or invalid
1416      ;      If EAX = FFFFFFFFh, it is 're-calculation needed'
1417      ;      sign due to r/w error
1418      ;
1419      ;      (Modified registers: eax, ebx, ecx, edx, esi)
1420      ;
1421      ;
1422 0000C561 66891D[1E810100] <1> mov [CFS_OPType], bx
1423 0000C568 A3[20810100] <1> mov [CFS_CC], eax
1424 <1>
1425 0000C56D 80FFFF <1> cmp bh, 0FFh
1426 0000C570 740B <1> je short pass_calculate_freespace_get_drive_dt_offset
1427 <1>
1428 loc_calculate_freespace_get_drive_dt_offset:
1429 <1> xor eax, eax
1430 0000C574 88FC <1> mov ah, bh
1431 0000C576 BE00010900 <1> mov esi, Logical_DOSDisks
1432 0000C57B 01C6 <1> add esi, eax
1433 <1>
1434 pass_calculate_freespace_get_drive_dt_offset:
1435 0000C57D 08DB <1> or bl, bl
1436 0000C57F 7436 <1> jz short loc_reset_fcc
1437 <1>
1438 loc_get_free_sectors:
1439 0000C581 8B4674 <1> mov eax, [esi+LD_FreeSectors]
1440 <1>
1441 <1> ;xor ecx, ecx
1442 <1> ;dec ecx ; 0FFFFFFFFh
1443 <1> ;cmp eax, ecx ; 29/02/2016
1444 <1> ;je short loc_get_free_sectors_retn ; recalculation is needed!
1445 <1>
1446 <1> ; 23/03/2016
1447 0000C584 8B4E70 <1> mov ecx, [esi+LD_TotalSectors]
1448 0000C587 39C1 <1> cmp ecx, eax ; Total sectors must be greater than Free sectors !
1449 0000C589 7707 <1> ja short loc_get_free_sectors_check_optype
1450 <1>
1451 0000C58B 31C0 <1> xor eax, eax
1452 0000C58D 48 <1> dec eax ; 0FFFFFFFFh ; recalculation is needed!
1453 0000C58E 894674 <1> mov [esi+LD_FreeSectors], eax ; reset (for recalculation)
1454 <1>
1455 loc_get_free_sectors_retn:
1456 0000C591 C3 <1> retn
1457 <1>
1458 loc_get_free_sectors_check_optype:
1459 0000C592 80FB03 <1> cmp bl, 3
1460 0000C595 7203 <1> jb short loc_set_fcc_1 ; 25/07/2022
1461 <1>
1462 0000C597 29C9 <1> sub ecx, ecx ; 0
1463 <1>
1464 0000C599 C3 <1> retn
1465 <1>
1466 loc_set_fcc_1:
1467 0000C59A 807E0302 <1> cmp byte [esi+LD_FATType], 2
1468 <1> ;ja loc_update_FAT32_fs_info_fcc
1469 <1> ; 25/07/2022
1470 0000C59E 7605 <1> jna short loc_set_fcc_2
1471 0000C5A0 E9DD000000 <1> jmp loc_update_FAT32_fs_info_fcc
1472 <1>
1473 loc_set_fcc_2:
1474 <1> ;mov eax, [esi+LD_FreeSectors]
1475 0000C5A5 0FB64E13 <1> movzx ecx, byte [esi+LD_BPB+SecPerClust]
1476 0000C5A9 29D2 <1> sub edx, edx
1477 0000C5AB F7F1 <1> div ecx
1478 <1> ;or dx, dx
1479 <1> ; DX -> Remain sectors < SecPerClust
1480 <1> ; DX > 0 -> invalid free sector count
1481 <1> ;jnz short loc_reset_fcc
1482 <1>
1483 ;pass_set_fcc_div32:
1484 0000C5AD A3[947E0100] <1> mov [FreeClusterCount], eax
1485 0000C5B2 E986000000 <1> jmp loc_set_free_sectors_FAT12_FAT16
1486 <1>
1487 loc_reset_fcc:
1488 0000C5B7 31C0 <1> xor eax, eax
1489 0000C5B9 A3[947E0100] <1> mov [FreeClusterCount], eax ; 0
1490 0000C5BE 8B5678 <1> mov edx, [esi+LD_Clusters]
1491 0000C5C1 42 <1> inc edx
1492 0000C5C2 8915[827E0100] <1> mov [LastCluster], edx
1493 <1>
1494 0000C5C8 807E0302 <1> cmp byte [esi+LD_FATType], 2
1495 0000C5CC 7645 <1> jna short loc_count_free_fat_clusters_0
1496 <1>
1497 0000C5CE 48 <1> dec eax ; FFFFFFFFh
1498 0000C5CF A3[28810100] <1> mov [CFS_FAT32FC], eax
1499 <1>
1500 <1> ; 29/02/2016
1501 0000C5D4 89463A <1> mov [esi+LD_BPB+BPB_Reserved], eax ; reset
1502 0000C5D7 89463E <1> mov [esi+LD_BPB+BPB_Reserved+4], eax ; reset
1503 <1>
1504 <1> ;mov eax, 2
1505 <1> ; 25/07/2022
1506 0000C5DA 40 <1> inc eax ; eax = 0
1507 0000C5DB B002 <1> mov al, 2
1508 <1>
1509 loc_count_fc_next_cluster_0:
1510 0000C5DD 50 <1> push eax
1511 0000C5DE E896F9FFFF <1> call get_next_cluster
1512 0000C5E3 7310 <1> jnc short loc_check_fat32_ff_cluster
1513 0000C5E5 09C0 <1> or eax, eax
1514 0000C5E7 741E <1> jz short pass_inc_cfs_fcc_0
1515 <1>
1516 loc_put_fcc_unknown_sign:
1517 0000C5E9 58 <1> pop eax
1518 <1> ; "Free count is Unknown" sign
1519 <1> ;mov dword [FreeClusterCount], 0FFFFFFFFh
1520 <1>
1521 <1> ; 29/02/2016
1522 <1> ; Save Free Cluster Count value in FAT32 'BPB_Reserved' area
1523 <1> ;mov [esi+LD_BPB+BPB_Reserved], 0FFFFFFFFh ; unknown!
1524 0000C5EA 8B15[28810100] <1> mov edx, [CFS_FAT32FC] ; First Free Cluster
1525 <1> ; Save First Free Cluster value in FAT32 'BPB_Reserved+4' area
1526 0000C5F0 89563E <1> mov [esi+LD_BPB+BPB_Reserved+4], edx
1527 <1>
1528 0000C5F3 EB7D <1> jmp loc_put_fcc_invalid_sign
1529 <1>
1530 loc_check_fat32_ff_cluster:
1531 0000C5F5 09C0 <1> or eax, eax

```

```

1532 0000C5F7 750E      <1>      jnz      short pass_inc_cfs_fcc_0
1533 0000C5F9 58       <1>      pop      eax
1534 0000C5FA A3[28810100] <1>      mov      [CFS_FAT32FC], eax
1535                               <1>      ;mov      dword [FreeClusterCount], 1
1536 0000C5FF FF05[947E0100] <1>      inc      dword [FreeClusterCount]
1537 0000C605 EB27      <1>      jmp      short pass_inc_cfs_fcc_1
1538                               <1>
1539                               <1> pass_inc_cfs_fcc_0:
1540 0000C607 58       <1>      pop      eax
1541                               <1>
1542                               <1> pass_inc_cfs_fcc_0c:
1543 0000C608 40       <1>      inc      eax ; add eax, 1
1544 0000C609 3B05[827E0100] <1>      cmp      eax, [LastCluster]
1545 0000C60F 76CC      <1>      jna      short loc_count_fc_next_cluster_0
1546 0000C611 EB6F      <1>      jmp      short loc_update_FAT32_fs_info_fcc
1547                               <1>
1548                               <1> loc_count_free_fat_clusters_0:
1549                               <1>      ;mov      eax, 2
1550 0000C613 B002      <1>      mov      al, 2
1551                               <1>
1552                               <1> loc_count_fc_next_cluster:
1553 0000C615 50       <1>      push     eax
1554 0000C616 E85EF9FFFF <1>      call     get_next_cluster
1555 0000C61B 720C      <1>      jc      short loc_count_fcc_stc
1556                               <1>
1557                               <1> loc_count_free_clusters_1:
1558 0000C61D 21C0      <1>      and      eax, eax
1559 0000C61F 750C      <1>      jnz      short pass_inc_cfs_fcc
1560                               <1>
1561 0000C621 FF05[947E0100] <1>      inc      dword [FreeClusterCount]
1562 0000C627 EB04      <1>      jmp      short pass_inc_cfs_fcc
1563                               <1>
1564                               <1> loc_count_fcc_stc:
1565 0000C629 09C0      <1>      or      eax, eax
1566 0000C62B 75BC      <1>      jnz      short loc_put_fcc_unknown_sign ; 29/02/2016
1567                               <1>
1568                               <1> pass_inc_cfs_fcc:
1569 0000C62D 58       <1>      pop      eax
1570                               <1>
1571                               <1> pass_inc_cfs_fcc_1:
1572 0000C62E 40       <1>      inc      eax ; add eax, 1
1573 0000C62F 3B05[827E0100] <1>      cmp      eax, [LastCluster]
1574 0000C635 76DE      <1>      jna      short loc_count_fc_next_cluster
1575                               <1>
1576                               <1> loc_set_free_sectors:
1577 0000C637 807E0302 <1>      cmp      byte [esi+LD_FATType], 2
1578 0000C63B 7745      <1>      ja      short loc_update_FAT32_fs_info_fcc
1579                               <1>
1580                               <1> loc_set_free_sectors_FAT12_FAT16:
1581 0000C63D 803D[1E810100]00 <1>      cmp      byte [CFS_OPType], 0
1582 0000C644 761C      <1>      jna      short pass_FAT_add_sub_fcc
1583 0000C646 A1[20810100] <1>      mov      eax, [CFS_CC]
1584 0000C648 803D[1E810100]01 <1>      cmp      byte [CFS_OPType], 1
1585 0000C652 7708      <1>      ja      short pass_FAT_add_fcc
1586 0000C654 0105[947E0100] <1>      add      [FreeClusterCount], eax
1587 0000C65A EB06      <1>      jmp      short pass_FAT_add_sub_fcc
1588                               <1>
1589                               <1> pass_FAT_add_fcc:
1590 0000C65C 2905[947E0100] <1>      sub      [FreeClusterCount], eax
1591                               <1>
1592                               <1> pass_FAT_add_sub_fcc:
1593 0000C662 0FB64613 <1>      movzx    eax, byte [esi+LD_BPB+SecPerClust]
1594 0000C666 8B15[947E0100] <1>      mov      edx, [FreeClusterCount]
1595 0000C66C F7E2      <1>      mul      edx
1596                               <1>
1597 0000C66E 31C9      <1>      xor      ecx, ecx
1598 0000C670 EB05      <1>      jmp      short loc_cfs_retn_params
1599                               <1>
1600                               <1> loc_put_fcc_invalid_sign:
1601 0000C672 29C0      <1>      sub      eax, eax ; 0
1602 0000C674 48       <1>      dec      eax ; FFFFFFFFh
1603                               <1> loc_fat32_ffc_recalc_needed:
1604 0000C675 89C1      <1>      mov      ecx, eax
1605                               <1>
1606                               <1> loc_cfs_retn_params:
1607 0000C677 894674 <1>      mov      [esi+LD_FreeSectors], eax
1608 0000C67A 0FB71D[1E810100] <1>      movzx    ebx, word [CFS_OPType]
1609 0000C681 C3       <1>      retn
1610                               <1>
1611                               <1> loc_update_FAT32_fs_info_fcc:
1612                               <1> loc_check_fcc_FSINFO_op:
1613                               <1>      ; 29/02/2016
1614                               <1>      ; EAX = Free cluster count (before this update) ; value from disk
1615                               <1>      ; EDX = First Free Cluster (before this update) ; value from disk
1616 0000C682 803D[1E810100]01 <1>      cmp      byte [CFS_OPType], 1
1617 0000C689 7221      <1>      jb      short loc_cfs_FAT32_get_rcalc_parms ; 0 = recalculated
1618 0000C68B 7406      <1>      je      short loc_check_fcc_FSINFO_op1 ; 1 = add
1619                               <1> loc_check_fcc_FSINFO_op2: ; subtract
1620 0000C68D F71D[20810100] <1>      neg      dword [CFS_CC] ; prepare to subtract ; 2 = sub (add negative)
1621                               <1> loc_check_fcc_FSINFO_op1:
1622                               <1>      ; 01/03/2016
1623 0000C693 31D2      <1>      xor      edx, edx ; 0
1624 0000C695 4A       <1>      dec      edx ; 0FFFFFFFFh
1625 0000C696 8B463A <1>      mov      eax, [esi+LD_BPB+BPB_Reserved]
1626 0000C699 39D0      <1>      cmp      eax, edx
1627                               <1>      ;jnb      short loc_put_fcc_invalid_sign
1628                               <1>      ; 21/12/2025
1629 0000C69B 73D8      <1>      jnb      short loc_fat32_ffc_recalc_needed
1630                               <1>
1631 0000C69D 0305[20810100] <1>      add      eax, [CFS_CC] ; free cluster count on disk + current count
1632 0000C6A3 72CD      <1>      jc      short loc_put_fcc_invalid_sign
1633                               <1>
1634 0000C6A5 A3[947E0100] <1>      mov      [FreeClusterCount], eax
1635 0000C6AA EB0E      <1>      jmp      short loc_cfs_write_FSINFO_sector
1636                               <1>
1637                               <1> loc_cfs_FAT32_get_rcalc_parms:
1638 0000C6AC 8B15[28810100] <1>      mov      edx, [CFS_FAT32FC]
1639 0000C6B2 A1[947E0100] <1>      mov      eax, [FreeClusterCount]
1640 0000C6B7 89563E <1>      mov      [esi+LD_BPB+BPB_Reserved+4], edx ; First Free Cluster
1641                               <1> loc_cfs_write_FSINFO_sector:
1642 0000C6BA 89463A <1>      mov      [esi+LD_BPB+BPB_Reserved], eax ; Free cluster count
1643                               <1>
1644 0000C6BD E8BC000000 <1>      call     set_fat32_fsinfo_sector_parms
1645 0000C6C2 72AE      <1>      jc      short loc_put_fcc_invalid_sign
1646                               <1>
1647                               <1> loc_set_FAT32_free_sectors:
1648                               <1>      ; 29/02/2016
1649                               <1>      ;mov      eax, [FreeClusterCount]
1650                               <1>      ;mov      ecx, eax
1651                               <1>      ;cmp      eax, 0FFFFFFFFh ; Invalid !
1652                               <1>      ;je      short loc_cfs_retn_params
1653                               <1>      ;
1654 0000C6C4 8B0D[947E0100] <1>      mov      ecx, [FreeClusterCount]
1655 0000C6CA 0FB64613 <1>      movzx    eax, byte [esi+LD_BPB+SecPerClust]

```

```

1656 0000C6CE F7E1      <1>      mul     ecx
1657                    <1>      ; 29/02/2016
1658 0000C6D0 31C9      <1>      xor     ecx, ecx ; 0
1659 0000C6D2 09D2      <1>      or      edx, edx ; 0 ?
1660 0000C6D4 759C      <1>      jnz     short loc_put_fcc_invalid_sign ; 25/07/2022
1661 0000C6D6 394670    <1>      cmp     [esi+LD_TotalSectors], eax ; Volume size in sectors
1662 0000C6D9 7697      <1>      jna     short loc_put_fcc_invalid_sign
1663                    <1>      ;
1664                    <1>      loc_set_FAT32_free_sectors_ok:
1665 0000C6DB 31D2      <1>      xor     edx, edx ; 0
1666 0000C6DD EB98      <1>      jmp     short loc_cfs_retn_params
1667                    <1>      ;
1668                    <1>      ;
1669                    <1>      get_last_cluster:
1670                    <1>      ; 22/10/2016
1671                    <1>      ; 27/02/2016 (TRDOS 386 = TRDOS v2.0)
1672                    <1>      ; 12/06/2010 (DRV_FAT.ASM, 'proc_get_last_custer')
1673                    <1>      ; 06/06/2010
1674                    <1>      ; INPUT ->
1675                    <1>      ; EAX = First Cluster Number
1676                    <1>      ; ESI = Logical Dos Drive Parameters Table
1677                    <1>      ; OUTPUT ->
1678                    <1>      ; cf = 0 -> No Error, EAX is valid
1679                    <1>      ; cf = 1 -> EAX > 0 -> Error
1680                    <1>      ; EAX = Last Cluster Number
1681                    <1>      ; ECX = Previous Cluster -just before the last cluster-
1682                    <1>      ; 22/10/2016
1683                    <1>      ; [glc_index] = cluster index number of the last cluster
1684                    <1>      ;
1685                    <1>      ; (Modified registers: EAX, ECX, EBX, EDX)
1686                    <1>      ;
1687 0000C6DF 89C1      <1>      mov     ecx, eax
1688                    <1>      ;
1689 0000C6E1 C705[30810100]FFFF- <1>      mov     dword [glc_index], 0FFFFFFFh ; 22/10/2016
1690 0000C6E9 FFFF      <1>      ;
1691                    <1>      loc_glc_get_next_cluster_1:
1692 0000C6EB 890D[2C810100]    <1>      mov     [glc_prevcluster], ecx
1693                    <1>      ; 22/10/2016
1694 0000C6F1 FF05[30810100] <1>      inc     dword [glc_index]
1695                    <1>      ;
1696                    <1>      loc_glc_get_next_cluster_2:
1697 0000C6F7 E87DF8FFFF    <1>      call    get_next_cluster
1698                    <1>      ; ecx = current/previous cluster
1699                    <1>      ; eax = next/last cluster
1700                    <1>      jnc     short loc_glc_get_next_cluster_1
1701                    <1>      ;
1702 0000C6FE 09C0      <1>      or      eax, eax
1703 0000C700 7509      <1>      jnz     short loc_glc_stc_retn
1704                    <1>      ;
1705                    <1>      ; ecx = previous cluster
1706 0000C702 89C8      <1>      mov     eax, ecx
1707                    <1>      ;
1708                    <1>      ; previous cluster becomes last cluster (ecx -> eax)
1709                    <1>      ; previous of previous cluster becomes previous cluster (ecx)
1710                    <1>      ;
1711                    <1>      loc_glc_prev_cluster_retn:
1712 0000C704 8B0D[2C810100] <1>      mov     ecx, [glc_prevcluster]
1713 0000C70A C3          <1>      retn
1714                    <1>      ;
1715                    <1>      loc_glc_stc_retn:
1716 0000C70B F5          <1>      cmc     ;stc
1717 0000C70C EBF6      <1>      jmp     short loc_glc_prev_cluster_retn
1718                    <1>      ;
1719                    <1>      truncate_cluster_chain:
1720                    <1>      ; 31/08/2024 - TRDOS 386 v2.0.9
1721                    <1>      ; 01/03/2016
1722                    <1>      ; 28/02/2016 (TRDOS 386 = TRDOS v2.0)
1723                    <1>      ; 22/01/2011 (DRV_FAT.ASM, 'proc_truncate_cluster_chain')
1724                    <1>      ; 11/09/2010
1725                    <1>      ; INPUT ->
1726                    <1>      ; ESI = Logical dos drive description table address
1727                    <1>      ; EAX = First cluster to be truncated/unlinked
1728                    <1>      ; OUTPUT ->
1729                    <1>      ; ESI = Logical dos drive description table address
1730                    <1>      ; ECX = Count of truncated/removed clusters
1731                    <1>      ; CF = 0 -> EAX = Free sectors
1732                    <1>      ; CF = 1 -> Error code in EAX (AL)
1733                    <1>      ;
1734                    <1>      ; NOTE: This procedure does not update lm date&time !
1735                    <1>      ;
1736                    <1>      loc_truncate_cc:
1737 0000C70E 31C9      <1>      xor     ecx, ecx ; mov ecx, 0
1738                    <1>      ;mov     byte [FAT_BuffValidData], 0
1739 0000C710 890D[7E7E0100] <1>      mov     [FAT_ClusterCounter], ecx ; 0 ; reset
1740                    <1>      ;
1741                    <1>      ;;;
1742                    <1>      ; 31/08/2024
1743 0000C716 21C0      <1>      and     eax, eax ; 0
1744 0000C718 740D      <1>      jz      short loc_tcc_unlink_zero_cluster ; zero
1745                    <1>      ;cmp     eax, 0FFFFFFFh
1746 0000C71A 3DFFFFFFF0F    <1>      cmp     eax, 0FFFFFFFh ; 28 bit cluster number limit (EOF)
1747 0000C71F 7207      <1>      jnb     short loc_tcc_unlink_clusters
1748                    <1>      ;
1749                    <1>      ; (possible FAT32) EOF signature...
1750                    <1>      ; not a valid cluster number
1751                    <1>      ;
1752                    <1>      ; NOTE: update_cluster returns EOF (if eax > [LastCluster])
1753                    <1>      ; instead of invalid data error
1754                    <1>      ;
1755 0000C721 B81D000000    <1>      mov     eax, ERR_INV_DATA ; invalid cluster number
1756 0000C726 F9          <1>      stc
1757                    <1>      loc_tcc_unlink_zero_cluster: ; nothing to do
1758 0000C727 C3          <1>      retn
1759                    <1>      ;;;
1760                    <1>      ;
1761                    <1>      loc_tcc_unlink_clusters:
1762 0000C728 E833FBFFFF    <1>      call    update_cluster
1763                    <1>      ; EAX = Next Cluster
1764                    <1>      ; ECX = Cluster Value
1765                    <1>      ; Note:
1766                    <1>      ; Returns count of unlinked clusters in
1767                    <1>      ; dword ptr FAT_ClusterCounter
1768 0000C72D 73F9      <1>      jnc     short loc_tcc_unlink_clusters
1769                    <1>      ;
1770                    <1>      ; error or EOF (end of cluster chain) ; 31/08/2024
1771                    <1>      ;
1772                    <1>      pass_tcc_unlink_clusters:
1773 0000C72F A2[37810100]    <1>      mov     byte [TCC_FATerr], al
1774 0000C734 803D[767E0100]02 <1>      cmp     byte [FAT_BuffValidData], 2
1775 0000C73B 750E      <1>      jne     short loc_tcc_calculate_FAT_freespace
1776 0000C73D E892FDFFFF    <1>      call    save_fat_buffer
1777 0000C742 7307      <1>      jnc     short loc_tcc_calculate_FAT_freespace
1778 0000C744 A2[37810100]    <1>      mov     byte [TCC_FATerr], al ; Error

```

```

1779      <1>      ;mov     byte [FAT_BuffValidData], 0
1780      <1>
1781      <1>      ; 01/03/2016
1782 0000C749 EB12      <1>      jmp     short loc_tcc_recalculate_FAT_freespace
1783      <1>
1784      <1> loc_tcc_calculate_FAT_freespace:
1785 0000C74B A1[7E7E0100] <1>      mov     eax, [FAT_ClusterCounter] ; signed (+-) number
1786 0000C750 66BB01FF <1>      mov     bx, 0FF01h ; BH = FFh -> ESI = Dos drv desc. table
1787      <1>      ; BL = 1 -> add cluster(s)
1788 0000C754 E808FEFFFF <1>      call    calculate_fat_freespace
1789 0000C759 21C9      <1>      and     ecx, ecx ; cx = 0 -> valid free sector count
1790 0000C75B 7409      <1>      jz      short pass_truncate_cc_recalc_FAT_freespace
1791      <1>
1792      <1> loc_tcc_recalculate_FAT_freespace:
1793 0000C75D 66BB00FF <1>      mov     bx, 0FF00h ; recalculate !
1794 0000C761 E8FBFDFFFF <1>      call    calculate_fat_freespace
1795      <1>
1796      <1> loc_tcc_calculate_FAT_freespace_err:
1797      <1> pass_truncate_cc_recalc_FAT_freespace:
1798 0000C766 8B0D[7E7E0100] <1>      mov     ecx, [FAT_ClusterCounter]
1799      <1>
1800 0000C76C 803D[37810100]00 <1>      cmp     byte [TCC_FATerr], 0
1801 0000C773 7608      <1>      jna     short loc_tcc_unlink_clusters_retn
1802      <1>
1803      <1> loc_tcc_unlink_clusters_error:
1804 0000C775 0FB605[37810100] <1>      movzx   eax, byte [TCC_FATerr]
1805 0000C77C F9      <1>      stc
1806      <1> loc_tcc_unlink_clusters_retn:
1807 0000C77D C3      <1>      retn
1808      <1>
1809      <1> set_fat32_fsinfo_sector_parms:
1810      <1>      ; 15/10/2016
1811      <1>      ; 23/03/2016
1812      <1>      ; 29/02/2016 (TRDOS 386 = TRDOS v2.0)
1813      <1>      ; INPUT ->
1814      <1>      ; ESI = Logical dos drive description table address
1815      <1>      ; [esi+LD_BPB+BPB_Reserved] = Free Cluster Count
1816      <1>      ; [esi+LD_BPB+BPB_Reserved+4] = First Free Cluster
1817      <1>      ; OUTPUT ->
1818      <1>      ; ESI = Logical dos drive description table address
1819      <1>      ; CF = 0 -> OK..
1820      <1>      ; CF = 1 -> Error code in EAX (AL)
1821      <1>      ;
1822      <1>      ; (Modified registers: EAX, EBX, ECX, EDX)
1823      <1>
1824 0000C77E E824000000 <1>      call    get_fat32_fsinfo_sector_parms
1825 0000C783 7221      <1>      jc      short update_fat32_fsinfo_sector_retn
1826      <1>
1827 0000C785 8B463A      <1>      mov     eax, [esi+LD_BPB+BPB_Reserved] ; Free Cluster Count
1828 0000C788 8B563E      <1>      mov     edx, [esi+LD_BPB+BPB_Reserved+4] ; First free Cluster
1829      <1>
1830      <1>      ;mov     ebx, DOSBootSectorBuff
1831 0000C78B 8983E8010000 <1>      mov     [ebx+488], eax
1832 0000C791 8993EC010000 <1>      mov     [ebx+492], edx
1833      <1>
1834 0000C797 A1[24810100] <1>      mov     eax, [CFS_FAT32FSINFOSEC]
1835 0000C79C B901000000 <1>      mov     ecx, 1
1836 0000C7A1 E82E520000 <1>      call    disk_write
1837      <1>      ;jnc     short update_fat32_fsinfo_sector_retn
1838      <1>
1839      <1>      ; 15/10/2016 (1dh -> 18)
1840      <1>      ; 23/03/2016 (1dh)
1841      <1>      ;mov     eax, 18 ; Drive not ready or write error
1842      <1>
1843      <1> update_fat32_fsinfo_sector_retn:
1844 0000C7A6 C3      <1>      retn
1845      <1>
1846      <1> get_fat32_fsinfo_sector_parms:
1847      <1>      ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
1848      <1>      ; 15/10/2016
1849      <1>      ; 23/03/2016
1850      <1>      ; 01/03/2016
1851      <1>      ; 29/02/2016 (TRDOS 386 = TRDOS v2.0)
1852      <1>      ; INPUT ->
1853      <1>      ; ESI = Logical dos drive description table address
1854      <1>      ; OUTPUT ->
1855      <1>      ; ESI = Logical dos drive description table address
1856      <1>      ; EBX = FSINFO sector buffer address (DOSBootSectorBuff)
1857      <1>      ; CF = 0 -> OK..
1858      <1>      ; EAX = FsInfo sector address
1859      <1>      ; ECX = Free cluster count
1860      <1>      ; EDX = First free cluster
1861      <1>      ; CF = 1 -> Error code in AL (EAX)
1862      <1>      ; EBX = 0
1863      <1>      ;
1864      <1>      ; [CFS_FAT32FSINFOSEC] = FAT32 FSINFO sector address
1865      <1>      ;
1866      <1>      ; (Modified registers: EAX, EBX, ECX, EDX)
1867      <1>
1868 0000C7A7 0FB74636 <1>      movzx   eax, word [esi+LD_BPB+FAT32_FSInfoSec]
1869 0000C7AB 03466C      <1>      add     eax, [esi+LD_StartSector]
1870 0000C7AE A3[24810100] <1>      mov     [CFS_FAT32FSINFOSEC], eax
1871      <1>
1872 0000C7B3 BB[727C0100] <1>      mov     ebx, DOSBootSectorBuff
1873      <1>      ;mov     ecx, 1
1874      <1>      ; 25/07/2022
1875 0000C7B8 29C9      <1>      sub     ecx, ecx
1876 0000C7BA FEC1      <1>      inc     cl
1877      <1>      ; ecx = 1
1878 0000C7BC E822520000 <1>      call    disk_read
1879 0000C7C1 722F      <1>      jc      short loc_read_FAT32_fsinfo_sec_err
1880      <1>
1881 0000C7C3 BB[727C0100] <1>      mov     ebx, DOSBootSectorBuff
1882      <1>
1883 0000C7C8 813B52526141 <1>      cmp     dword [ebx], 41615252h
1884 0000C7CE 751E      <1>      jne     short loc_read_FAT32_fsinfo_sec_stc
1885      <1>
1886 0000C7D0 81BBE4010000727241- <1>      cmp     dword [ebx+484], 61417272h
1887 0000C7D9 61      <1>      jne     short loc_read_FAT32_fsinfo_sec_stc
1888      <1>
1889 0000C7DC A1[24810100] <1>      mov     eax, [CFS_FAT32FSINFOSEC]
1890 0000C7E1 8B8BE8010000 <1>      mov     ecx, [ebx+488] ; free cluster count
1891 0000C7E7 8B93EC010000 <1>      mov     edx, [ebx+492] ; first (next) free cluster
1892      <1>
1893 0000C7ED C3      <1>      retn
1894      <1>
1895      <1> loc_read_FAT32_fsinfo_sec_stc:
1896      <1>      ; 15/10/2016 (0Bh -> 28)
1897      <1>      ;mov     eax, 28 ; Invalid format!
1898      <1>      ; 25/07/2022
1899 0000C7EE B31C      <1>      mov     bl, 28
1900 0000C7F0 EB02      <1>      jmp     short loc_read_FAT32_fsinfo_sec_stc_retn
1901      <1>

```

```

1902 <1> loc_read_FAT32_fsinfo_sec_err:
1903 <1> ; 15/10/2016 (15h -> 17)
1904 <1> ; 23/03/2016 (15h)
1905 <1> ;mov eax, 17 ; Drive not ready or read error
1906 <1> ; 25/07/2022
1907 0000C7F2 B311 <1> mov bl, 17
1908 <1> loc_read_FAT32_fsinfo_sec_stc_retn:
1909 <1> ; 25/07/2022
1910 0000C7F4 29C0 <1> sub eax, eax
1911 0000C7F6 88D8 <1> mov al, bl ; error code
1912 <1> ; eax = error code
1913 0000C7F8 29DB <1> sub ebx, ebx ; 0
1914 0000C7FA F9 <1> stc
1915 0000C7FB C3 <1> retn
1916 <1>
1917 <1> add_new_cluster:
1918 <1> ; 30/08/2024
1919 <1> ; 27/08/2024
1920 <1> ; 25/08/2024 - TRDOS 386 v2.0.9
1921 <1> ; 25/07/2022 (TRDOS 386 kernel v2.0.5)
1922 <1> ; 15/10/2016
1923 <1> ; 16/05/2016
1924 <1> ; 18/03/2016, 24/03/2016
1925 <1> ; 11/03/2016 (TRDOS 386 = TRDOS v2.0)
1926 <1> ; 30/07/2011 (DRV_FAT.ASM)
1927 <1> ; 11/09/2010
1928 <1> ; INPUT ->
1929 <1> ; ESI = Logical dos drv desc. table address
1930 <1> ; EAX = Last cluster
1931 <1> ; OUTPUT ->
1932 <1> ; ESI = Logical dos drv desc. table address
1933 <1> ; EAX = New Last cluster (next cluster)
1934 <1> ; cf = 1 -> error code in EAX (AL)
1935 <1> ; cf = 1 -> EBX = sectors per cluster
1936 <1> ; ECX = Free sectors
1937 <1> ;;; 25/07/2022
1938 <1> ; (EBX = sectors per cluster -not used-)
1939 <1> ; EDX = 0 (if cf = 0)
1940 <1> ; NOTE:
1941 <1> ; This procedure does not update lm date&time !
1942 <1> ; 30/08/2024
1943 <1> ; and doesn't update 1st clust and file size fields !
1944 <1> ;
1945 <1> ; (Modified registers: EAX, EBX, ECX, EDX, EDI)
1946 <1>
1947 0000C7FC A3[54820100] <1> mov [FAT_anc_LCcluster], eax
1948 <1>
1949 0000C801 E885F9FFFF <1> call get_first_free_cluster
1950 0000C806 7206 <1> jc short loc_add_new_cluster_retn
1951 <1> ; EAX >= 2 and EAX < FFFFFFFFh is valid
1952 <1>
1953 <1> ;mov edx, eax
1954 <1> ;
1955 <1> ;inc edx
1956 <1> ;jnz short loc_add_new_cluster_check_ffc_eax
1957 <1> ;jnz short loc_add_new_cluster_save_ffc
1958 <1>
1959 <1> ; 27/08/2024
1960 0000C808 40 <1> inc eax ; (*)
1961 0000C809 750D <1> jnz short loc_add_new_cluster_save_ffc
1962 <1>
1963 <1> loc_add_new_cluster_no_disk_space_retn:
1964 <1> ;mov eax, 27h ; MSDOS err => insufficient disk space
1965 <1> ; 27/08/2024
1966 <1> ; eax = 0
1967 <1> ; 25/07/2022
1968 <1> ;xor eax, eax
1969 0000C80B B027 <1> mov al, 27h
1970 <1> loc_add_new_cluster_stc_retn:
1971 0000C80D F9 <1> stc
1972 <1> loc_add_new_cluster_retn:
1973 <1> ; 25/07/2022
1974 <1> ;movzx ebx, byte [esi+LD_BPB+SecPerClust]
1975 0000C80E 8B4E74 <1> mov ecx, [esi+LD_FreeSectors]
1976 <1> ;xor edx, edx
1977 <1> ;stc
1978 0000C811 C3 <1> retn
1979 <1>
1980 <1> loc_anc_invalid_format_stc_retn:
1981 <1> ; 27/08/2024
1982 <1> ;stc
1983 <1> loc_add_new_cluster_invalid_format_retn:
1984 <1> ; 15/10/2016 (0Bh -> 28)
1985 <1> ;mov eax, 28 ; Invalid format
1986 <1> ;jmp short loc_add_new_cluster_retn
1987 <1> ; 25/07/2022
1988 0000C812 29C0 <1> sub eax, eax
1989 0000C814 B01C <1> mov al, 28
1990 0000C816 EBF5 <1> jmp short loc_add_new_cluster_stc_retn
1991 <1>
1992 <1> ;loc_add_new_cluster_check_ffc_eax:
1993 <1> ; cmp eax, 2
1994 <1> ; jb short loc_add_new_cluster_invalid_format_retn
1995 <1>
1996 <1> loc_add_new_cluster_save_ffc:
1997 <1> ;;;
1998 <1> ; 27/08/2024
1999 0000C818 48 <1> dec eax ; (*)
2000 <1> ;;;
2001 0000C819 A3[58820100] <1> mov [FAT_anc_FFcluster], eax
2002 <1>
2003 <1> ; 27/08/2024 (TRDOS 386 v2.0.9)
2004 <1> %if 0
2005 <1> sub eax, 2
2006 <1> movzx ebx, byte [esi+LD_BPB+SecPerClust]
2007 <1> mul ebx
2008 <1> or edx, edx
2009 <1> jnz short loc_anc_invalid_format_stc_retn
2010 <1>
2011 <1> loc_add_new_cluster_allocate_cluster:
2012 <1> ; 18/03/2016
2013 <1> xchg edx, eax ; eax = 0
2014 <1> ; 16/05/2016
2015 <1> ;cmp [ClusterBuffer_Valid], al ; 0
2016 <1> ;jna short loc_anc_clear_cluster_buffer
2017 <1> ;; 'copy' command,
2018 <1> ;; writing destination file clust after reading source file clust
2019 <1> ;mov [ClusterBuffer_Valid], al ; 0 ; reset
2020 <1> ;jmp short loc_add_new_cluster_write_nc_to_disk
2021 <1>
2022 <1> loc_anc_clear_cluster_buffer:
2023 <1> ; 11/03/2016
2024 <1> ; Clear buffer
2025 <1> mov edi, Cluster_Buffer ; 70000h (for current TRDOS 386 version)

```

```

2026      mov     ecx, ebx ; sector count
2027      shl     ecx, 7 ; 1 sector = 512 bytes -> 128 double words
2028      ;xor     eax, eax ; 0
2029      rep     stosd
2030
2031      <1> loc_add_new_cluster_write_nc_to_disk:
2032      <1> ; 11/03/2016
2033      <1> ;xchg     eax, edx ; edx = 0, eax = sector offset
2034      <1> mov     eax, edx
2035      <1> add     eax, [esi+LD_DATABegin]
2036      <1> jc     short loc_add_new_cluster_invalid_format_retn
2037
2038      <1> mov     ecx, ebx ; ECX = sectors per cluster (<256)
2039      <1> mov     ebx, Cluster_Buffer
2040      <1> call    disk_write
2041      <1> jnc     short loc_add_new_cluster_update_fat_nlc
2042
2043      <1> ; 15/10/2016 (1Dh -> 18)
2044      <1> ;mov     eax, 18 ; Write Error
2045      <1> ; 25/07/2022
2046      <1> xor     eax, eax
2047      <1> mov     al, 18
2048      <1> jmp     short loc_add_new_cluster_stc_retn
2049
2050      <1> loc_add_new_cluster_update_fat_nlc:
2051      <1> mov     eax, [FAT_anc_FFcluster]
2052      <1> %endif
2053      <1> ; 30/08/2024
2054      <1> ; eax = [FAT_anc_FFcluster] ; first free cluster
2055      <1> xor     ecx, ecx ; 0
2056      <1> mov     [FAT_ClusterCounter], ecx ; 0 ; reset
2057      <1> dec     ecx ; -1 ; 0FFFFFFFh
2058      <1> test    [FAT_anc_LCcluster], ecx ; 0 ?
2059      <1> jz     short loc_add_new_cluster_update_fat_fc ; yes
2060
2061      <1> ; 27/08/2024
2062      <1> ; eax = (first free) cluster to be added as last cluster
2063      <1> ;;;
2064      <1> ;xor     ecx, ecx
2065
2066      <1> loc_add_new_cluster_update_fat_nlc: ; 30/08/2024
2067      <1> ;mov     [FAT_ClusterCounter], ecx ; 0 ; reset
2068      <1> ;dec     ecx ; 0FFFFFFFh ; last cluster
2069      <1> call    update_cluster
2070      <1> jnc     short loc_add_new_cluster_update_fat_plc
2071      <1> or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
2072      <1> jnz     short loc_add_new_cluster_stc_retn
2073
2074      <1> loc_add_new_cluster_update_fat_plc:
2075      <1> mov     eax, [FAT_anc_LCcluster]
2076      <1> mov     ecx, [FAT_anc_FFcluster]
2077      <1> loc_add_new_cluster_update_fat_fc: ; 30/08/2024
2078      <1> call    update_cluster
2079      <1> jnc     short loc_add_new_cluster_save_fat_buffer
2080
2081      <1> or     eax, eax ; EAX = 0 -> cluster value is 0 or eocc
2082      <1> jz     short loc_add_new_cluster_save_fat_buffer
2083
2084      <1> loc_anc_save_fat_buffer_err_retn:
2085      <1> ;cmp     byte [FAT_ClusterCounter], 1
2086      <1> ;jb     short loc_add_new_cluster_retn
2087
2088      <1> mov     bx, 0FF00h ; recalculate free space (BL = 0)
2089      <1> ; (BH = FFh -> Use ESI as Drv Param. Tbl.)
2090      <1> push    eax
2091      <1> call    calculate_fat_freespace
2092      <1> pop     eax
2093      <1> jmp     loc_add_new_cluster_stc_retn
2094
2095      <1> loc_add_new_cluster_save_fat_buffer:
2096      <1> ;cmp     byte [FAT_BuffValidData], 2
2097      <1> ;jne     short loc_add_new_cluster_calc_FAT_freespace
2098      <1> ;Byte [FAT_BuffValidData] = 2
2099      <1> call    save_fat_buffer
2100      <1> jc     short loc_anc_save_fat_buffer_err_retn
2101
2102      <1> loc_add_new_cluster_calc_FAT_freespace:
2103      <1> ;mov     eax, 1 ; Only one Cluster
2104      <1> mov     eax, [FAT_ClusterCounter]
2105      <1> mov     bx, 0FF01h ; BH = FFh -> ESI -> Dos drv desc. table
2106      <1> ; BL = 1 -> add cluster(s)
2107      <1> ;mov     bl, 01h ; 27/08/2024
2108      <1> ; NOTE: EAX value will be added to Free Cluster Count
2109      <1> ; (Free Cluster Count is decreased when EAX value is negative)
2110      <1> call    calculate_fat_freespace
2111      <1> ; ECX = 0 -> no error, ECX > 0 -> error or invalid return
2112      <1> and     ecx, ecx ; ECX = 0 -> valid free sector count
2113      <1> jz     short loc_add_new_cluster_return_cluster_number
2114
2115      <1> loc_add_new_cluster_recalc_FAT_freespace:
2116      <1> mov     bx, 0FF00h ; recalculate free space
2117      <1> call    calculate_fat_freespace
2118      <1> ; cf = 0
2119      <1> loc_add_new_cluster_return_cluster_number:
2120      <1> mov     ecx, eax ; Free sector count
2121      <1> mov     eax, [FAT_anc_FFcluster]
2122      <1> mov     edi, Cluster_Buffer
2123      <1> ; 25/07/2022 (EBX is not used by callers of this sprocedure)
2124      <1> ;movzx   ebx, byte [esi+LD_BPB+SecPerClust]
2125      <1> xor     edx, edx ; 0
2126      <1> retn
2127
2128      <1> write_cluster:
2129      <1> ; 19/12/2025 - TRDOS 386 v2.0.10
2130      <1> ; 31/08/2024 - TRDOS 386 v2.0.9
2131      <1> ; 15/10/2016
2132      <1> ; 21/03/2016 (TRDOS 386 = TRDOS v2.0)
2133      <1> ;
2134      <1> ; INPUT ->
2135      <1> ; EAX = Cluster Number (Sector index for SINGLIX FS)
2136      <1> ; ESI = Logical DOS Drive Description Table address
2137      <1> ; EBX = Cluster (File R/W) Buffer address (max. 64KB)
2138      <1> ; Only for SINGLIX FS:
2139      <1> ; EDX = File Number (The 1st FDT address)
2140      <1> ; OUTPUT ->
2141      <1> ; cf = 1 -> Cluster can not be written onto disk
2142      <1> ; EAX > 0 -> Error number
2143      <1> ; cf = 0 -> Cluster has been written successfully
2144      <1> ;
2145      <1> ; (Modified registers: EAX, ECX, EBX, EDX)
2146      <1>
2147      <1> movzx   ecx, byte [esi+LD_BPB+BPB_SecPerClust]
2148      <1> ; CL = 1 = [esi+LD_FS_Reserved2] ; SectPerClust for Singlix FS
2149      <1>

```

```

2150 <1> write_file_sectors: ; 16/03/2016
2151 <1>
2152 <1> ; 19/12/2025
2153 <1> %if 0
2154 <1>     cmp     byte [esi+LD_FATType], 0
2155 <1>     jna     short write_fs_cluster
2156 <1> %endif
2157 <1>
2158 <1> write_fat_file_sectors:
2159 <1>     ; 31/08/2024
2160 <1>     ; ecx = sector count (may be different than sectors per cluster)
2161 <1>     sub     eax, 2 ; Beginning cluster number is always 2
2162 <1>     movzx   edx, byte [esi+LD_BPB+BPB_SecPerClust] ; 18/03/2016
2163 <1>     mul     edx
2164 <1>     add     eax, [esi+LD_DATABegin] ; absolute address of the cluster
2165 <1>
2166 <1>     ; EAX = Disk sector address
2167 <1>     ; ECX = Sector count
2168 <1>     ; EBX = Buffer address
2169 <1>     ; (EDX = 0)
2170 <1>     ; ESI = Logical DOS drive description table address
2171 <1>
2172 <1>     call    disk_write
2173 <1>     jnc     short wclust_retn
2174 <1>
2175 <1>     ; 15/10/2016 (1Dh -> 18)
2176 <1>     mov     eax, 18 ; Drive not ready or write error !
2177 <1>     retn
2178 <1>
2179 <1> wclust_retn:
2180 <1>     sub     eax, eax ; 0
2181 <1>     retn
2182 <1>
2183 <1> ; 19/12/2025
2184 <1> %if 0
2185 <1>
2186 <1> write_fs_cluster:
2187 <1>     ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
2188 <1>     ; 21/03/2016 (TRDOS 386 = TRDOS v2.0)
2189 <1>     ; Singlix FS
2190 <1>
2191 <1>     ; EAX = Cluster number is sector index number of the file (eax)
2192 <1>
2193 <1>     ; EDX = File number is the first File Descriptor Table address
2194 <1>     ; of the file. (Absolute address of the FDT).
2195 <1>
2196 <1>     ; eax = sector index (0 for the first sector)
2197 <1>     ; edx = FDT0 address
2198 <1>     ; 64 KB buffer = 128 sectors (limit)
2199 <1>     ; mov     ecx, 128 ; maximum count of sectors (before eof)
2200 <1>     ; 25/07/2022
2201 <1>     sub     ecx, ecx
2202 <1>     mov     cl, 128
2203 <1>     ; call    write_fs_sectors
2204 <1>     ; retn
2205 <1>     ; jmp     short write_fs_sectors
2206 <1>
2207 <1> write_fs_sectors:
2208 <1>     ; 21/03/2016 (TRDOS 386 = TRDOS v2.0)
2209 <1>     stc
2210 <1>     retn
2211 <1>
2212 <1> %endif
2213 <1>
2214 <1> get_cluster_by_index:
2215 <1>     ; 19/12/2025 (TRDOS 386 v2.0.10)
2216 <1>     ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
2217 <1>     ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
2218 <1>     ; INPUT ->
2219 <1>     ;     EAX = Beginning cluster
2220 <1>     ;     EDX = Sector index in disk/file section
2221 <1>     ;     (Only for SINGLIX file system!)
2222 <1>     ;     ECX = Cluster sequence number after the beginning cluster
2223 <1>     ;     ESI = Logical DOS Drive Description Table address
2224 <1>     ; OUTPUT ->
2225 <1>     ;     EAX = Cluster number
2226 <1>     ;     cf = 1 -> Error code in AL (EAX)
2227 <1>     ;
2228 <1>     ; (Modified registers: EAX, ECX, EBX, EDX)
2229 <1>
2230 <1> ; 19/12/2025
2231 <1> %if 0
2232 <1>     cmp     byte [esi+LD_FATType], 1
2233 <1>     jnb     short get_fs_section_by_index
2234 <1> %endif
2235 <1>     cmp     ecx, [esi+LD_Clusters]
2236 <1>     jnb     short gcbi_1
2237 <1> gcbi_0:
2238 <1>     ; stc
2239 <1>     ; mov     eax, 23h ; Cluster not available !
2240 <1>     ;         ; MSDOS error code: FCB unavailable
2241 <1>     ; 25/07/2022
2242 <1>     sub     eax, eax
2243 <1> gcbi_4:
2244 <1>     mov     al, 23h
2245 <1>     stc
2246 <1>     retn
2247 <1> gcbi_1:
2248 <1>     push    ecx
2249 <1>     call    get_next_cluster
2250 <1>     pop     ecx
2251 <1>     jc      short gcbi_3
2252 <1>     loop    gcbi_1
2253 <1> gcbi_2:
2254 <1>     retn
2255 <1> gcbi_3:
2256 <1>     or      eax, eax
2257 <1>     ; jz      short gcbi_0
2258 <1>     ; 25/07/2022
2259 <1>     jz      short gcbi_4
2260 <1>     cmc     ; stc
2261 <1>     retn
2262 <1>
2263 <1> ; 19/12/2025
2264 <1> %if 0
2265 <1>
2266 <1> get_fs_section_by_index:
2267 <1>     ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
2268 <1>     ; INPUT ->
2269 <1>     ;     EAX = Beginning FDT number/address
2270 <1>     ;     EDX = Sector index in disk/file section
2271 <1>     ;     ECX = Sector sequence number after the beginning FDT
2272 <1>     ;     ESI = Logical DOS Drive Description Table address
2273 <1>     ; OUTPUT ->

```

```

2274      ;      EAX = FDT number/address
2275      ;      EDX = Sector index of the section (0,1,2,3,4...)
2276      ;      cf = 1 -> Error code in AL (EAX)
2277      ;
2278      ;(Modified registers: EAX, ECX, EBX, EDX)
2279      ;
2280      mov     eax, 0FFFFFFFh
2281      retn
2282
2283      get_last_section:
2284      ;      22/10/2016 (TRDOS 386 = TRDOS v2.0)
2285      ;      INPUT ->
2286      ;      EAX = (The 1st) FDT number/address
2287      ;      ESI = Logical DOS Drive Description Table address
2288      ;      OUTPUT ->
2289      ;      EAX = FDT number/address of the last section
2290      ;      EDX = Last sector of the section (0,1,2,3,4...)
2291      ;      [glc_index] = sector index number of the last sector
2292      ;      (for file, not for the last section)
2293      ;
2294      ;      cf = 1 -> Error code in AL (EAX)
2295      ;
2296      ;(Modified registers: EAX, ECX, EBX, EDX)
2297      ;
2298      mov     eax, 0
2299      mov     edx, 0
2300      retn
2301
2302      %endif
2303
2304      %include 'trdosk6.s' ; 24/01/2016
2305      ; *****
2306      ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.10) - MAIN PROGRAM : trdosk6.s
2307      ; *****
2308      ; Last Update: 10/01/2026 (Previous: 27/09/2024, v2.0.9)
2309      ; *****
2310      ; Beginning: 24/01/2016
2311      ; *****
2312      ; Assembler: NASM version 2.15 (trdos386.s)
2313      ; *****
2314      ; Derived from 'Retro UNIX 386 Kernel - v0.2.1.0' source code by Erdogan Tan
2315      ; u1.s (27/17/2015), u2.s (03/01/2016)
2316      ; *****
2317      ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
2318      ; TRDOS2.ASM (09/11/2011)
2319      ; *****
2320      ; INT_21H.ASM (c) 2009-2011 Erdogan TAN [14/11/2009] Last Update: 08/11/2011
2321      ; *****
2322      ; Ref: Retro UNIX 386 v1.2 Kernel (v0.2.2.3) - ux.s - 15/07/2022
2323
2324      sysent: ; < enter to system call >
2325      ; 23/07/2022 - TRDOS 386 Kernel v2.0.5
2326      ; 17/03/2017
2327      ; 03/03/2017
2328      ; 19/02/2017
2329      ; 13/01/2017
2330      ; 06/06/2016
2331      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
2332      ; 16/04/2015 - 19/10/2015 (Retro UNIX 386 v1)
2333      ; 10/04/2013 - 18/01/2014 (Retro UNIX 8086 v1)
2334      ;
2335      ; 'unkni' or 'sysent' is sytem entry from various traps.
2336      ; The trap type is determined and an indirect jump is made to
2337      ; the appropriate system call handler. If there is a trap inside
2338      ; the system a jump to panic is made. All user registers are saved
2339      ; and u.sp points to the end of the users stack. The sys (trap)
2340      ; instructor is decoded to get the the system code part (see
2341      ; trap instruction in the PDP-11 handbook) and from this
2342      ; the indirect jump address is calculated. If a bad system call is
2343      ; made, i.e., the limits of the jump table are exceeded, 'badsys'
2344      ; is called. If the call is legitimate control passes to the
2345      ; appropriate system routine.
2346      ;
2347      ; Calling sequence:
2348      ; Through a trap caused by any sys call outside the system.
2349      ; Arguments:
2350      ; Arguments of particular system call.
2351      ; .....
2352      ; Retro UNIX 8086 v1 modification:
2353      ; System call number is in EAX register.
2354      ;
2355      ; Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
2356      ; registers depending of function details.
2357      ;
2358      ; 16/04/2015
2359      mov     [ss:u.sp], esp ; Kernel stack points to return address
2360
2361      ; save user registers
2362      push    ds
2363      push    es
2364      push    fs
2365      push    gs
2366      pushad  ; eax, ecx, edx, ebx, esp -before pushad-, ebp, esi, edi
2367      ;
2368      ; ESPACE = [ss:u.sp] - esp ; 4*12 = 48 ; 17/09/2015 ; 06/06/2016
2369      ; (ESPACE is size of space in kernel stack
2370      ; for saving/restoring user registers.)
2371      ;
2372      push    eax ; 01/07/2015
2373      mov     ax, KDATA
2374      mov     ds, ax
2375      mov     es, ax
2376      mov     fs, ax
2377      mov     gs, ax
2378      mov     eax, [k_page_dir]
2379      mov     cr3, eax
2380      pop     eax ; 01/07/2015
2381      ; 19/10/2015
2382      cld
2383      ;
2384      inc     byte [sysflg]
2385      ; incb sysflg / indicate a system routine is in progress
2386      sti     ; 18/01/2014
2387      ;jnz     panic ; 24/05/2013
2388      ; beq     1f
2389      ; jmp     panic ; / called if trap inside system
2390      ; 23/07/2022
2391      jz      short sysent0
2392      jmp     panic
2393
2394      ;1:
2395      sysent0:
2396      ; 17/03/2017
2397      and     byte [esp+ESPACE+8], ~1 ; clear carry flag
2398

```



```

95      <1>      ; 16/04/2015
96 0000C8FE A3[548E0100] <1>      mov     [u.r0], eax
97 0000C903 8925[508E0100] <1>      mov     [u.usp], esp ; kernel stack points to user's registers
98      <1>
99      <1>      ; 13/01/2017 (TRDOS 386 Feaure only !)
100 0000C909 803D[CC8E0100]00 <1>      cmp     byte [u.t_lock], 0 ; timer interrupt lock ?
101      <1>      ;ja     sysrele      ; yes, sys release only !!!
102      <1>      ; 23/07/2022
103 0000C910 7605 <1>      jna     short sysent1
104 0000C912 E902020000 <1>      jmp     sysrele
105      <1>      ; mov $s.syst+2,clockp
106      <1>      ; mov r0,-(sp) / save user registers
107      <1>      ; mov sp,u.r0 / pointer to bottom of users stack
108      <1>      ; / in u.r0
109      <1>      ; mov r1,-(sp)
110      <1>      ; mov r2,-(sp)
111      <1>      ; mov r3,-(sp)
112      <1>      ; mov r4,-(sp)
113      <1>      ; mov r5,-(sp)
114      <1>      ; mov ac,-(sp) / "accumulator" register for extended
115      <1>      ; / arithmetic unit
116      <1>      ; mov mq,-(sp) / "multiplier quotient" register for the
117      <1>      ; / extended arithmetic unit
118      <1>      ; mov sc,-(sp) / "step count" register for the extended
119      <1>      ; / arithmetic unit
120      <1>      ; mov sp,u.sp / u.sp points to top of users stack
121      <1>      ; mov 18.(sp),r0 / store pc in r0
122      <1>      ; mov -(r0),r0 / sys inst in r0
123      <1>      ; sub $sys,r0 / get xxx code
124      <1>      sysent1:
125 0000C917 C1E002 <1>      shl     eax, 2
126      <1>      ; asl r0 / multiply by 2 to jump indirect in bytes
127 0000C91A 3DBC000000 <1>      cmp     eax, end_of_syscalls - syscalls
128      <1>      ; cmp r0,$2f-1f / limit of table (35) exceeded
129      <1>      ;jnb     short badsys
130      <1>      ; bhis badsys / yes, bad system call
131 0000C91F F5 <1>      cmc
132 0000C920 9C <1>      pushf
133 0000C921 50 <1>      push     eax
134 0000C922 8B2D[4C8E0100] <1>      mov     ebp, [u.sp] ; kernel stack at the beginning of sys call
135 0000C928 B0FE <1>      mov     al, 0FEh ; 11111110b
136 0000C92A 1400 <1>      adc     al, 0 ; al = al + cf
137 0000C92C 204508 <1>      and     [ebp+8], al ; flags (reset carry flag)
138      <1>      ; bic $341,20.(sp) / set users processor priority to 0
139      <1>      ; / and clear carry bit
140 0000C92F 5D <1>      pop     ebp ; eax
141 0000C930 9D <1>      popf
142 0000C931 720B <1>      jc      short badsys ; 23/07/2022
143 0000C933 A1[548E0100] <1>      mov     eax, [u.r0]
144      <1>      ; system call registers: EAX, EDX, ECX, EBX, ESI, EDI
145 0000C938 FFA5[8BC90000] <1>      jmp     dword [ebp+syscalls]
146      <1>      ; jmp *1f(r0) / jump indirect thru table of addresses
147      <1>      ; / to proper system routine.
148      <1>
149      <1>      ; 20/08/2024 (exit code)
150      <1>      ; 30/07/2022
151      <1>      ; 23/07/2022
152      <1>      badsys:
153      <1>      ; 25/12/2016
154      <1>      ; 18/04/2016 (TRDOS 386 = TRDOS v2.0)
155      <1>      ; 17/04/2011 (TRDOS v1.0, 'IFC.ASM')
156      <1>      ; 03/02/2011 ('trdos_ifc_routine')
157      <1>      ;
158      <1>      ; 16/04/2015 (Retro UNIX 386 v1, 'badsys')
159      <1>      ; (EIP, EAX values will be shown on screen with error message)
160      <1>      ; (EIP = 'CD 40h' instruction address -INT 40h-)
161      <1>      ; (EAX = Function number)
162      <1>      ;
163 0000C93E FE05[A48E0100] <1>      inc     byte [u.bsys]
164      <1>      ;
165 0000C944 8B1D[4C8E0100] <1>      mov     ebx, [u.sp] ; esp at the beginning of 'sysent'
166 0000C94A 8B03 <1>      mov     eax, [ebx] ; EIP (return address, not 'INT 30h' address)
167      <1>      ;sub     eax, 2 ; CDh, ##h
168      <1>      ; 30/07/2022
169 0000C94C 48 <1>      dec     eax
170 0000C94D 48 <1>      dec     eax
171 0000C94E E8A878FFFF <1>      call    dwordtohex
172 0000C953 8915[94350100] <1>      mov     [eip_str], edx
173 0000C959 A3[98350100] <1>      mov     [eip_str+4], eax
174 0000C95E A1[548E0100] <1>      mov     eax, [u.r0]
175 0000C963 E89378FFFF <1>      call    dwordtohex
176 0000C968 8915[83350100] <1>      mov     [eax_str], edx
177 0000C96E A3[87350100] <1>      mov     [eax_str+4], eax
178      <1>
179 0000C973 66C705[78350100]34- <1>      mov     word [int_num_str], SYSCALL_INT_NUM ; 25/12/2016
180 0000C97B 30 <1>
181 0000C97C BE[4A350100] <1>      mov     esi, ifc_msg ; "invalid function call !" msg (trdosk9.s)
182      <1>
183 0000C981 E824A4FFFF <1>      call    print_msg
184      <1>
185      <1>      ;; 20/08/2024 - temporary
186      <1>      ; mov     ebx, 07h
187      <1>      ; mov     ah, 0Eh
188      <1>      ;p_fc_msg:
189      <1>      ; lodsb
190      <1>      ; and     al, al
191      <1>      ; jz      short p_fc_ms_ok
192      <1>      ; call    _int10h
193      <1>      ; jmp     short p_fc_msg
194      <1>      ;p_fc_ms_ok:
195      <1>
196      <1>      ;jmp     sysexit
197      <1>
198      <1>      ; 20/08/2024 (exit code)
199      <1>      ;mov     bl, 0FFh ; -1
200      <1>      ;jmp     sysexit
201      <1>      ;
202 0000C986 E92C020000 <1>      jmp     sysexit_0FFh
203      <1>
204      <1>      syscalls: ; 1:
205      <1>      ; 20/08/2024 - TRDOS 386 v2.0.9
206      <1>      ; 31/12/2017
207      <1>      ; 28/02/2017
208      <1>      ; 20/02/2017
209      <1>      ; 19/02/2017
210      <1>      ; 15/10/2016
211      <1>      ; 20/05/2016
212      <1>      ; 19/05/2016
213      <1>      ; 16/05/2016
214      <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
215      <1>      ; 21/09/2015
216      <1>      ; 01/07/2015
217      <1>      ; 16/04/2015 (32 bit address modification)

```

```

218 0000C98B [570A0100] <1> dd sysver ; 0 ; Get TRDOS 386 version number (v2.0)
219 0000C98F [BCCB0000] <1> dd sysexit ; 1
220 0000C993 [E4CD0000] <1> dd sysfork ; 2
221 0000C997 [B1D10000] <1> dd sysread ; 3
222 0000C99B [D0D10000] <1> dd syswrite ; 4
223 0000C99F [CECF0000] <1> dd sysopen ; 5
224 0000C9A3 [87D10000] <1> dd sysclose ; 6
225 0000C9A7 [48CD0000] <1> dd syswait ; 7
226 0000C9AB [F1CE0000] <1> dd syscreat ; 8
227 0000C9AF [43170100] <1> dd sysrename ; 9 ; TRDOS 386, Rename File (31/12/2017)
228 0000C9B3 [F0120100] <1> dd sysdelete ; 10 ; TRDOS 386, Delete File (29/12/2017)
229 0000C9B7 [7EFE0000] <1> dd sysexec ; 11
230 0000C9BB [20140100] <1> dd syschdir ; 12
231 0000C9BF [AD150100] <1> dd systime ; 13 ; TRDOS 386, Get Sys Date&Time (30/12/2017)
232 0000C9C3 [4AD10000] <1> dd sysmkdir ; 14
233 0000C9C7 [54140100] <1> dd syschmod ; 15 ; TRDOS 386, Change Attributes (30/12/2017)
234 0000C9CB [57130100] <1> dd sysrmdir ; 16 ; TRDOS 386, Remove Directory (29/12/2017)
235 0000C9CF [06030100] <1> dd sysbreak ; 17
236 0000C9D3 [02150100] <1> dd sysdrive ; 18 ; TRDOS 386, Get/Set Current Drv (30/12/2017)
237 0000C9D7 [75030100] <1> dd sysseek ; 19
238 0000C9DB [8C030100] <1> dd systell ; 20
239 0000C9DF [4E180100] <1> dd sysmem ; 21 ; TRDOS 386, Get Total&Free Mem (31/12/2017)
240 0000C9E3 [84180100] <1> dd sysprompt ; 22 ; TRDOS 386, Change Cmd Prompt (31/12/2017)
241 0000C9E7 [C5180100] <1> dd syspath ; 23 ; TRDOS 386, Get/Set Run Path (31/12/2017)
242 0000C9EB [2A190100] <1> dd sysenv ; 24 ; TRDOS 386, Get/Set Env Vars (31/12/2017)
243 0000C9EF [2E160100] <1> dd sysstime ; 25 ; TRDOS 386, Set Sys Date&Time (30/12/2017)
244 0000C9F3 [EB030100] <1> dd sysquit ; 26
245 0000C9F7 [D4030100] <1> dd sysintr ; 27
246 0000C9FB [51150100] <1> dd sysdir ; 28 ; TRDOS 386, Get Curr Drive&Dir (30/12/2017)
247 0000C9FF [47CA0000] <1> dd sysemt ; 29
248 0000CA03 [8C150100] <1> dd sysldrvt ; 30 ; TRDOS 386, Get Logical DOS DDT (30/12/2017)
249 0000CA07 [CBD30000] <1> dd sysvideo ; 31 ; TRDOS 386 Video Functions (16/05/2016)
250 0000CA0B [52200100] <1> dd sysaudio ; 32 ; TRDOS 386 Audio Functions (16/05/2016)
251 0000CA0F [36D20000] <1> dd systimer ; 33 ; TRDOS 386 Timer Functions (18/05/2016)
252 0000CA13 [07040100] <1> dd sysssleep ; 34 ; Retro UNIX 8086 v1 feature only !
253 <1> ; 11/06/2014
254 0000CA17 [47040100] <1> dd sysmsg ; 35 ; Retro UNIX 386 v1 feature only !
255 <1> ; 01/07/2015
256 0000CA1B [62050100] <1> dd sysgeterr ; 36 ; Retro UNIX 386 v1 feature only !
257 <1> ; 21/09/2015 - get last error number
258 0000CA1F [C7120100] <1> dd sysfpstat ; 37 ; TRDOS 386 FPU state option (28/02/2017)
259 0000CA23 [660A0100] <1> dd syspri ; 38 ; change priority - TRDOS 386 (20/05/2016)
260 0000CA27 [19CB0000] <1> dd sysrele ; 39 ; TRDOS 386 (19/05/2016) (0 -> 39)
261 0000CA2B [910B0100] <1> dd sysfff ; 40 ; Find First File - TRDOS 386 (15/10/2016)
262 0000CA2F [680C0100] <1> dd sysfnf ; 41 ; Find Next File - TRDOS 386 (15/10/2016)
263 0000CA33 [B5110100] <1> dd sysalloc ; 42 ; Allocate contiguous memory block/pages
264 <1> ; TRDOS 386 (19/02/2017) DMA buff fuctions
265 0000CA37 [6F120100] <1> dd sysdalloc ; 43 ; Deallocate contiguous memory block/pages
266 <1> ; TRDOS 386 (19/02/2017) DMA buff fuctions
267 0000CA3B [AA120100] <1> dd syscalbac ; 44 ; IRQ Callback and Signal Response Byte
268 <1> ; service setup - TRDOS 386 (20/02/2017)
269 <1> ; 28/08/2017 (20/08/2017)
270 0000CA3F [33290100] <1> dd sysdma ; 45 ; TRDOS 386 - (ISA) DMA service
271 0000CA43 [6F2C0100] <1> dd sysstdio ; 46 ; TRDOS 386 v2.0.9 (STDIN/STDOUT functions)
272 <1>
273 <1> end_of_syscalls:
274 <1>
275 <1> sysemt: ; enable (or disable) multi tasking -time sharing-
276 <1>
277 <1> ; 08/08/2022
278 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
279 <1> ; 23/05/2016 - TRDOS 386 (TRDOS v2.0)
280 <1> ; 14/05/2015 (Retro UNIX 386 v1)
281 <1> ; 10/12/2013 - 20/04/2014 (Retro UNIX 8086 v1)
282 <1>
283 <1> ; Retro UNIX 8086 v1 modification:
284 <1> ; 'Enable Multi Tasking' system call instead
285 <1> ; of 'Emulator Trap' in original UNIX v1 for PDP-11.
286 <1>
287 <1> ; Retro UNIX 8086 v1 feature only!
288 <1> ; Using purpose: kernel will start without time-out
289 <1> ; (internal clock/timer) functionality.
290 <1> ; Then etc/init will enable clock/timer for
291 <1> ; multi tasking.
292 <1>
293 <1> ; INPUT ->
294 <1> ; BL = 0 -> disable multi tasking
295 <1> ; BL > 1 -> enable multi tasking (time sharing)
296 <1> ; OUTPUT ->
297 <1> ; none
298 <1>
299 <1> ; Note: Multi tasking is disabled during system
300 <1> ; initialization, it must be enabled by using
301 <1> ; this system call. (Otherwise, running proces
302 <1> ; will not be changed by another process within
303 <1> ; run time sequence/schedule, if running process
304 <1> ; will not 'release' itself. Only 'wakeup' procedure
305 <1> ; for waiting processes and programmed timer events
306 <1> ; for other processes can change running process
307 <1> ; while multi tasking is disabled.) ** 23/05/2016 **
308 <1>
309 0000CA47 803D[A68E0100]00 <1> cmp byte [u.uid], 0 ; root ?
310 <1> ;ja short error
311 <1> ; 23/07/2022
312 <1> ;ja short badsys ; 14/05/2015
313 <1> ; 08/08/2022
314 0000CA4E 7605 <1> jna short sysemt_root
315 0000CA50 E9E9FEFFFF <1> jmp badsys
316 <1> sysemt_root: ; 08/08/2022
317 0000CA55 FA <1> cli
318 0000CA56 881D[32830100] <1> mov [multi_tasking], bl ; 0 to disable, >0 to enable
319 0000CA5C EB20 <1> jmp sysret
320 <1>
321 <1> error:
322 <1> ; 18/05/2016
323 <1> ; 13/05/2016
324 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
325 <1> ; 16/04/2015 - 17/09/2015 (Retro UNIX 386 v1)
326 <1> ; 10/04/2013 - 07/08/2013 (Retro UNIX 8086 v1)
327 <1>
328 <1> ; 'error' merely sets the error bit off the processor status (c-bit)
329 <1> ; then falls right into the 'sysret', 'sysrele' return sequence.
330 <1>
331 <1> ; INPUTS -> none
332 <1> ; OUTPUTS ->
333 <1> ; processor status - carry (c) bit is set (means error)
334 <1>
335 <1> ; 26/05/2013 (Stack pointer must be reset here!
336 <1> ; Because, jumps to error procedure
337 <1> ; disrupts push-pop nesting balance)
338 <1>
339 0000CA5E 8B2D[4C8E0100] <1> mov ebp, [u.sp] ; interrupt (system call) return (iretd) address
340 0000CA64 804D0801 <1> or byte [ebp+8], 1 ; set carry bit of flags register
341 <1> ; (system call will return with cf = 1)

```

```

342          ; bis $1,20.(r1) / set c bit in processor status word below
343          ; / users stack
344          ; 17/09/2015
345 0000CA68 83ED30      sub     ebp, ESPACE ; 48 ; total size of stack frame ('sysdefs.inc')
346          ; / for saving/restoring user registers
347          ; cmp     ebp, [u.usp]
348          ; je      short err0
349 0000CA6B 892D[508E0100]  mov     [u.usp], ebp
350          ; err0:
351          ; 01/09/2015
352 0000CA71 8B25[508E0100]  mov     esp, [u.usp] ; Retro Unix 8086 v1 modification!
353          ; 10/04/2013
354          ; (If an I/O error occurs during disk I/O,
355          ; related procedures will jump to 'error'
356          ; procedure directly without returning to
357          ; the caller procedure. So, stack pointer
358          ; must be restored here.)
359          ; 13/05/2016
360          ; NOTE: (The last) error code is in 'u.error', it can be retrieved by
361          ; 'get last error' system call later.
362          ;
363          ; 03/09/2015 - 09/06/2015 - 07/08/2013
364 0000CA77 C605[BA8E0100]00  mov     byte [u.kcall], 0 ; namei_r, mkdir_w reset
365          ;
366          ; sysret: ; < return from system call >
367          ; 28/08/2024 - TRDOS 386 kernel v2.0.9
368          ; 23/07/2022 - TRDOS 386 kernel v2.0.5
369          ; 01/03/2017
370          ; 28/02/2017
371          ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
372          ; 16/04/2015 - 10/09/2015 (Retro UNIX 386 v1)
373          ; 10/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
374          ;
375          ; 'sysret' first checks to see if process is about to be
376          ; terminated (u.bsys). If it is, 'sysexit' is called.
377          ; If not, following happens:
378          ; 1) The user's stack pointer is restored.
379          ; 2) r1=0 and 'iget' is called to see if last mentioned
380          ; i-node has been modified. If it has, it is written out
381          ; via 'ppoke'.
382          ; 3) If the super block has been modified, it is written out
383          ; via 'ppoke'.
384          ; 4) If the dismountable file system's super block has been
385          ; modified, it is written out to the specified device
386          ; via 'ppoke'.
387          ; 5) A check is made if user's time quantum (uquant) ran out
388          ; during his execution. If so, 'tswap' is called to give
389          ; another user a chance to run.
390          ; 6) 'sysret' now goes into 'sysrele'.
391          ; (See 'sysrele' for conclusion.)
392          ;
393          ; Calling sequence:
394          ; jump table or 'br sysret'
395          ; Arguments:
396          ; -
397          ; .....
398          ; ((AX=r1 for 'iget' input))
399          ;
400          ;
401 0000CA7E 31C0      xor     eax, eax ; 28/02/2017
402          ; sysret0: ; 29/07/2015 (eax = 0, jump from sysexec)
403 0000CA80 FEC0      inc     al ; 04/05/2013
404 0000CA82 3805[A48E0100]  cmp     [u.bsys], al ; 1
405          ; ; tstb u.bsys / is a process about to be terminated because
406          ; jnb     sysexit ; 04/05/2013
407          ; ; bne sysexit / of an error? yes, go to sysexit
408          ; 23/07/2022
409 0000CA88 7205      jb      short sysret1
410          ; jmp     sysexit
411          ; 22/08/2024
412 0000CA8A E92B010000  jmp     sysexit_@ ; BL = 0FFh ; -1
413          ; sysret1: ; 23/07/2022
414          ; mov     esp, [u.usp] ; 24/05/2013 (that is not needed here)
415          ; mov     u.sp, sp / no point stack to users stack
416 0000CA8F FEC8      dec     al ; mov ax, 0
417          ; ; clr r1 / zero r1 to check last mentioned i-node
418 0000CA91 E83D4F0000  call    iget
419          ; ; jsr r0,iget / if last mentioned i-node has been modified
420          ; ; / it is written out
421          ; 10/01/2017
422          ; 09/01/2017
423          ; sysrele: ; < release >
424          ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
425          ; 16/04/2015 - 14/10/2015 (Retro UNIX 386 v1)
426          ; 10/04/2013 - 07/03/2014 (Retro UNIX 8086 v1)
427          ;
428          ; 'sysrele' first calls 'tswap' if the time quantum for a user is
429          ; zero (see 'sysret'). It then restores the user's registers and
430          ; turns off the system flag. It then checked to see if there is
431          ; an interrupt from the user by calling 'isintr'. If there is,
432          ; the output gets flashed (see isintr) and interrupt action is
433          ; taken by a branch to 'intract'. If there is no interrupt from
434          ; the user, a rti is made.
435          ;
436          ; Calling sequence:
437          ; Fall through a 'bne' in 'sysret' & ?
438          ; Arguments:
439          ; -
440          ; .....
441          ;
442          ; 23/02/2014 (swapret)
443          ; 22/09/2013
444          ; sysrele0: ; 1:
445 0000CA96 803D[9C8E0100]00  cmp     byte [u.quant], 0 ; 16/05/2013
446          ; ; tstb uquant / is the time quantum 0?
447 0000CA9D 7705      ja      short swapret
448          ; ; bne 1f / no, don't swap it out
449          ; sysrele: ; 07/12/2013 (jump from 'clock')
450 0000CA9F E87C3E0000  call    tswap
451          ; ; jsr r0,tswap / yes, swap it out
452          ;
453          ; Retro Unix 8086 v1 feature: return from 'swap' to 'swapret' address.
454          ; swapret: ; 1:
455          ; 10/09/2015
456          ; 01/09/2015
457          ; 14/05/2015
458          ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit, pm modifications)
459          ; 26/05/2013 (Retro UNIX 8086 v1)
460          ; cli
461          ; 24/07/2015
462          ;
463          ; ; 'esp' must be already equal to '[u.usp]' here !
464          ; ; mov esp, [u.usp]
465          ;

```

```

466      ; 22/09/2013
467 0000CAA4 E8234F0000      <1>      call    isintr
468      ; 20/10/2013
469 0000CAA9 7405          <1>      jz      short sysrel1
470 0000CAAB E8F4000000      <1>      call    intract
471      ; jsr r0, isintr / is there an interrupt from the user
472      ; br intract / yes, output gets flushed, take interrupt
473      ; / action
474      <1>      sysrel1:
475 0000CAB0 FA            <1>      cli      ; 14/10/2015
476      <1>      sysrel2:
477      ; 28/02/2017
478      ; Check if there is a (delayed) callback for current user/process
479 0000CAB1 A0[CF8E0100]    <1>      mov     al, [u.irqwait]
480 0000CAB6 240F          <1>      and     al, 0Fh ; is there a waiting IRQ callback service ?
481 0000CAB8 7444          <1>      jz      short sysrel8 ; no
482      <1>
483      ; Set return to IRQ callback service and return from the service
484 0000CABA 0FB6D8          <1>      movzx   ebx, al
485 0000CABD 883D[CF8E0100]    <1>      mov     [u.irqwait], bh ; 0 ; reset
486 0000CAC3 8A9B[A0350100]    <1>      mov     bl, [ebx+IRQenum] ; (available) IRQ index +1 (1 to 9)
487      ; 01/03/2017
488 0000CAC9 FECB          <1>      dec     bl ; IRQ index number, 0 to 8
489 0000CACB 7831          <1>      js      short sysrel8 ; 0 -> FFh (not in use!?)
490      <1>
491 0000CACD A0[A58E0100]    <1>      mov     al, [u.uno] ; current process (user) number
492 0000CAD2 3883[80880100]    <1>      cmp     [ebx+IRQ.owner], al
493 0000CAD8 7524          <1>      jne     short sysrel8 ; it is not the current user/process !?
494 0000CADA F683[92880100]01 <1>      test    byte [ebx+IRQ.method], 1 ; callback ?
495 0000CAE1 741B          <1>      jz      short sysrel8 ; not a callback method !?
496      <1>
497 0000CAE3 8B93[A4880100]    <1>      mov     edx, [ebx+IRQ.addr] ; IRQ callback service address (virtual)
498 0000CAE9 C605[D08E0100]01 <1>      mov     byte [u.r_lock], 1 ; IRQ callback service in progress flag
499      <1>
500 0000CAF0 E8D23E0000      <1>      call    wswap ; save user's registers & status
501      <1>      ; (for return from IRQ callback service)
502      <1>
503 0000CAF5 8B2D[4C8E0100]    <1>      mov     ebp, [u.sp] ; kernel's stack, points to EIP (user)
504 0000CAFB 895500          <1>      mov     [ebp], edx ; IRQ call back service address
505      <1>      sysrel8:
506 0000CAFE FE0D[488E0100]    <1>      dec     byte [sysflg]
507      <1>      ; decb sysflg / turn system flag off
508      <1>
509 0000CB04 A1[AC8E0100]    <1>      mov     eax, [u.pgdir]
510 0000CB09 0F22D8          <1>      mov     cr3, eax ; 1st PDE points to Kernel Page Table 0 (1st 4 MB)
511      <1>      ; (others are different than kernel page tables)
512      ; 10/09/2015
513 0000CB0C 61            <1>      popad ; edi, esi, ebp, temp (increment esp by 4), ebx, edx, ecx, eax
514      <1>      ; mov (sp)+,sc / restore user registers
515      <1>      ; mov (sp)+,mq
516      <1>      ; mov (sp)+,ac
517      <1>      ; mov (sp)+,r5
518      <1>      ; mov (sp)+,r4
519      <1>      ; mov (sp)+,r3
520      <1>      ; mov (sp)+,r2
521      <1>
522 0000CB0D A1[548E0100]    <1>      mov     eax, [u.r0] ; ((return value in EAX))
523 0000CB12 0FA9          <1>      pop     gs
524 0000CB14 0FA1          <1>      pop     fs
525 0000CB16 07            <1>      pop     es
526 0000CB17 1F            <1>      pop     ds
527      <1>      ; or word [esp+8], 200h ; 22/01/2017 ; force enabling interrupts
528 0000CB18 CF            <1>      iretd
529      <1>      ; rti / no, return from interrupt
530      <1>
531      <1>      sysrele:
532      ; 23/07/2022 - TRDOS 386 v2.0.5
533      ; 24/03/2017
534      ; 28/02/2017
535      ; 27/02/2017
536      ; 29/01/2017
537      ; 14/01/2017
538      ; 13/01/2017
539      ; 09/01/2017 - 10/01/2017 - 12/01/2017
540      ; Major modification for TRDOS 386 (Callback return)
541      ;
542      ; 'sysrele' system call restores previously saved
543      ; registers and addresses of the process
544      ; (Main purpose -in TRDOS 386- is to return from
545      ; timer callback service routine in ring 3 -user mode-.)
546      ;
547      ; check if the process is in timer callback phase
548 0000CB19 803D[CC8E0100]00 <1>      cmp     byte [u.t_lock], 0 ; TIMER INT LOCK
549      <1>      ; je short sysrel0 ; classic (Retro UNIX 386 type) sysrele
550 0000CB20 7735          <1>      ja      short sysrel3
551      <1>      ; 27/02/2017
552 0000CB22 803D[D08E0100]00 <1>      cmp     byte [u.r_lock], 0 ; IRQ callback lock
553      <1>      ; jna sysrel0 ; classic sysrele ; 24/03/2017
554      <1>      ; 23/07/2022
555 0000CB29 7705          <1>      ja      short sysrel9
556 0000CB2B E966FFFFFF      <1>      jmp     sysrel0
557      <1>      sysrel9:
558      ; 23/07/2022
559 0000CB30 E859000000      <1>      call    sysrel7
560 0000CB35 803D[D08E0100]00 <1>      cmp     byte [u.r_lock], 0 ; IRQ callback service lock
561 0000CB3C 7628          <1>      jna      short sysrel4
562 0000CB3E C605[D08E0100]00 <1>      mov     byte [u.r_lock], 0 ; reset
563      <1>      ; mov byte [u.irqwait], 0 ; reset ; 28/02/2017
564 0000CB45 A0[D18E0100]    <1>      mov     al, [u.r_mode]
565 0000CB4A 08C0          <1>      or      al, al
566 0000CB4C 7518          <1>      jnz     short sysrel4
567 0000CB4E FEC8          <1>      dec     al
568 0000CB50 A2[D18E0100]    <1>      mov     [u.r_mode], al ; 0FFh ; not necessary !?
569 0000CB55 EB32          <1>      jmp     short sysrel6
570      <1>      sysrel3:
571      ; 27/02/2017
572 0000CB57 E832000000      <1>      call    sysrel7
573      <1>      ; 14/01/2017
574 0000CB5C 28C0          <1>      sub     al, al
575 0000CB5E 3805[CC8E0100]    <1>      cmp     [u.t_lock], al ; 0 ; TIMER INT LOCK
576 0000CB64 770E          <1>      ja      short sysrel5 ; yes
577      <1>      sysrel4:
578      ; 29/01/2017
579 0000CB66 8B44241C          <1>      mov     eax, [esp+28] ; eax
580 0000CB6A A3[548E0100]    <1>      mov     [u.r0], eax
581 0000CB6F E93DFFFFFF      <1>      jmp     sysrel2
582      <1>      sysrel5:
583 0000CB74 A2[CC8E0100]    <1>      mov     [u.t_lock], al ; 0 ; reset
584 0000CB79 A0[CD8E0100]    <1>      mov     al, [u.t_mode]
585      <1>      and     al, al
586      <1>      ; jnz short sysrel2 ; 0FFh ; user mode
587 0000CB80 75E4          <1>      jnz     short sysrel4 ; 29/01/2017
588 0000CB82 FEC8          <1>      dec     al
589 0000CB84 A2[CD8E0100]    <1>      mov     [u.t_mode], al ; 0FFh ; not necessary !?
590      <1>      sysrel6:

```

```

590      <1>      ; cpu will continue from the interrupted sytem call addr
591 0000CB89 61      <1>      popad      ; edi, esi, ebp, esp, ebx, edx, ecx, eax
592 0000CB8A 83C410 <1>      add      esp, 16; pass segment segisters: ds, es, fs, gs
593 0000CB8D CF      <1>      iretd     ; eip, cs, eflags
594      <1>
595      <1> sysrel7:
596 0000CB8E 0FB61D[A58E0100] <1>      movzx   ebx, byte [u.uno] ; current process number
597      <1>      ;shl     bx, 2
598      <1>      ; 23/07/2022
599 0000CB95 C1E302 <1>      shl     ebx, 2
600      <1>      ;cmp    [ebx+p.tcb-4], eax ; 0 ; is there callback address ?
601      <1>      ;jna     short sysrel0
602      <1>      ; yes, reset callback address then restore process registers
603      <1>      ;mov     [ebx+p.tcb-4], eax ; 0 ; reset
604 0000CB98 8B83[948D0100] <1>      mov     eax, [ebx+p.upage-4] ; UPAGE address
605 0000CB9E FA      <1>      cli      ; disable interrupts till 'iretd'
606 0000CB9F E95B3E0000 <1>      jmp     rswap ; restore process 'u' structure
607      <1>
608      <1> intract: ; / interrupt action
609      <1>      ; 14/10/2015
610      <1>      ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
611      <1>      ; 09/05/2013 - 07/12/2013 (Retro UNIX 8086 v1)
612      <1>      ;
613      <1>      ; Retro UNIX 8086 v1 modification !
614      <1>      ; (Process/task switching and quit routine by using
615      <1>      ; Retro UNIX 8086 v1 keyboard interrupt output.))
616      <1>      ;
617      <1>      ; input -> 'u.quit' (also value of 'u.intr' > 0)
618      <1>      ; output -> If value of 'u.quit' = FFFFh ('ctrl+brk' sign)
619      <1>      ; 'intract' will jump to 'sysexit'.
620      <1>      ; Intract will return to the caller
621      <1>      ; if value of 'u.quit' <> FFFFh.
622      <1>      ; 14/10/2015
623 0000CBA4 FB      <1>      sti
624      <1>      ; 07/12/2013
625 0000CBA5 66FF05[A28E0100] <1>      inc     word [u.quit]
626 0000CBAC 7408      <1>      jz      short intrct0 ; FFFFh -> 0
627 0000CBAE 66FF0D[A28E0100] <1>      dec     word [u.quit]
628      <1>      ; 16/04/2015
629 0000CBB5 C3      <1>      retn
630      <1> intrct0:
631 0000CBB6 58      <1>      pop     eax ; call intract -> retn
632      <1>      ;
633      <1>      ; 20/08/2024
634      <1>      ;xor     eax, eax
635      <1>      ;inc     al ; mov ax, 1
636      <1>      ;;;
637      <1>      ; UNIX v1 original 'intract' routine...
638      <1>      ; / interrupt action
639      <1>      ;cmp     *(sp), $rti / are you in a clock interrupt?
640      <1>      ; bne 1f / no, 1f
641      <1>      ; cmp (sp)+, (sp)+ / pop clock pointer
642      <1>      ; 1: / now in user area
643      <1>      ; mov r1, -(sp) / save r1
644      <1>      ; mov u.ttyp, r1
645      <1>      ; / pointer to tty buffer in control-to r1
646      <1>      ; cmpb 6(r1), $177
647      <1>      ; / is the interrupt char equal to "del"
648      <1>      ; beq 1f / yes, 1f
649      <1>      ; clrb 6(r1)
650      <1>      ; / no, clear the byte
651      <1>      ; / (must be a quit character)
652      <1>      ; mov (sp)+, r1 / restore r1
653      <1>      ; clr u.quit / clear quit flag
654      <1>      ; bis $20, 2(sp)
655      <1>      ; / set trace for quit (sets t bit of
656      <1>      ; / ps-trace trap)
657      <1>      ; rti ; / return from interrupt
658      <1>      ; 1: / interrupt char = del
659      <1>      ; clrb 6(r1) / clear the interrupt byte
660      <1>      ; / in the buffer
661      <1>      ; mov (sp)+, r1 / restore r1
662      <1>      ; cmp u.intr, $core / should control be
663      <1>      ; / transferred to loc core?
664      <1>      ; blo 1f
665      <1>      ; jmp *u.intr / user to do rti yes,
666      <1>      ; / transfer to loc core
667      <1>      ; 1:
668      <1>      ; sys 1 / exit
669      <1>
670      <1>      ; 20/08/2024
671      <1>      ; ctrl+break -> exit code = -1
672      <1> sysexit_0FFh:
673      <1>      ; 20/08/2024
674 0000CBB7 31C0      <1>      xor     eax, eax
675 0000CBB9 40      <1>      inc     eax ; mov eax, 1
676      <1> sysexit_@: ; 22/08/2024
677 0000CBBB B3FF      <1>      mov     bl, 0FFh ; exit code ; -1
678      <1>
679      <1> sysexit: <terminate process>
680      <1>      ; 08/01/2026 - TRDOS 386 v2.0.10
681      <1>      ; 22/08/2024
682      <1>      ; 20/08/2024
683      <1>      ; 18/08/2024 - TRDOS 386 v2.0.9
684      <1>      ; 30/07/2022
685      <1>      ; 23/07/2022 - TRDOS 386 v2.0.5
686      <1>      ; 14/11/2017
687      <1>      ; 27/05/2017
688      <1>      ; 10/04/2017
689      <1>      ; 26/02/2017 - 28/02/2017
690      <1>      ; 02/01/2017 - 23/01/2017
691      <1>      ; 06/06/2016 - 10/06/2016
692      <1>      ; 19/05/2016 - 23/05/2016
693      <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
694      <1>      ; 16/04/2015 - 01/09/2015 (Retro UNIX 386 v1)
695      <1>      ; 19/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
696      <1>      ;
697      <1>      ; 'sysexit' terminates a process. First each file that
698      <1>      ; the process has opened is closed by 'flose'. The process
699      <1>      ; status is then set to unused. The 'p.pid' table is then
700      <1>      ; searched to find children of the dying process. If any of
701      <1>      ; children are zombies (died by not waited for), they are
702      <1>      ; set free. The 'p.pid' table is then searched to find the
703      <1>      ; dying process's parent. When the parent is found, it is
704      <1>      ; checked to see if it is free or it is a zombie. If it is
705      <1>      ; one of these, the dying process just dies. If it is waiting
706      <1>      ; for a child process to die, it notified that it doesn't
707      <1>      ; have to wait anymore by setting it's status from 2 to 1
708      <1>      ; (waiting to active). It is awakened and put on runq by
709      <1>      ; 'putlu'. The dying process enters a zombie state in which
710      <1>      ; it will never be run again but stays around until a 'wait'
711      <1>      ; is completed by it's parent process. If the parent is not
712      <1>      ; found, process just dies. This means 'swap' is called with
713      <1>      ; 'u.uno=0'. What this does is the 'wswap' is not called

```

```

714      <1>      ; to write out the process and 'rswap' reads the new process
715      <1>      ; over the one that dies..i.e., the dying process is
716      <1>      ; overwritten and destroyed.
717      <1>      ;
718      <1>      ; Calling sequence:
719      <1>      ;     sysexit or conditional branch.
720      <1>      ; Arguments:
721      <1>      ;     -
722      <1>      ; .....
723      <1>      ;
724      <1>      ; Retro UNIX 8086 v1 modification:
725      <1>      ;     System call number (=1) is in EAX register.
726      <1>      ;
727      <1>      ;     Other parameters are in EDX, EBX, ECX, ESI, EDI, EBP
728      <1>      ;     registers depending of function details.
729      <1>      ;
730      <1>      ; ('swap' procedure is mostly different than original UNIX v1.)
731      <1>      ;
732      <1>      ; / terminate process
733      <1>      ; AX = 1
734      <1>      ; dec ax ; 0
735      <1>      ; 18/08/2024
736      0000CBB8 48      <1>      dec     eax
737      0000CBB9 66A3[A08E0100] <1>      mov     [u.intr], ax ; 0
738      <1>      ; clr u.intr / clear interrupt control word
739      <1>      ; clr r1 / clear r1
740      <1>      ; 18/08/2024 - exit code (in EBX, BL)
741      0000CBC3 881D[D28E0100] <1>      mov     [u.exit], bl
742      <1>      sysexit_0:
743      <1>      ; 08/01/2026 (BugFix)
744      <1>      ; 30/07/2022
745      <1>      ; 23/01/2017
746      <1>      ; 02/01/2017
747      <1>      ; 10/06/2016
748      <1>      ; 06/06/2016
749      <1>      ; 23/05/2016
750      <1>      ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
751      <1>      ; Check and stop/clear timer event(s) of this (dying) process
752      <1>      ; if there is.
753      <1>      ;
754      <1>      ; 02/01/2017
755      0000CBC9 FA      <1>      cli     ; disable interrupts
756      <1>      ; 23/01/2017 - reset timer frequency (to 18.2Hz)
757      0000CBCA B036      <1>      mov     al, 00110110b ; 36h
758      0000CBCB E643      <1>      out     43h, al
759      0000CBCD 28C0      <1>      sub     al, al ; 0
760      0000CBDE E640      <1>      out     40h, al ; LB
761      0000CBDF E640      <1>      out     40h, al ; HB
762      <1>      ;
763      0000CBDD 0FB61D[A58E0100] <1>      movzx   ebx, byte [u.uno]
764      <1>      ; mov     bl, [u.uno] ; process number of dying process
765      0000CBDB 3883[D78D0100] <1>      cmp     byte [ebx+p.timer-1], al ; 0
766      0000CBE1 7639      <1>      jna     short sysexit_12 ; no timer events for this process
767      0000CBE3 8883[D78D0100] <1>      mov     byte [ebx+p.timer-1], al ; 0 ; reset
768      <1>      ; mov     al, [timer_events]
769      <1>      ; or      al, al
770      <1>      ; jz     short sysexit_12 ; no timer events
771      <1>      ; mov     cl, al
772      0000CBE9 8A0D[33830100] <1>      mov     cl, [timer_events] ; 14/11/2017
773      <1>      ; cli     ; disable interrupts
774      0000CBEF B410      <1>      mov     ah, 16 ; number of available timer events
775      0000CBF1 BE[688F0100] <1>      mov     esi, timer_set ; beginning address of timer events
776      <1>      sysexit_7:
777      0000CBF6 8A06      <1>      mov     al, [esi] ; process number (of timer event)
778      0000CBF8 38D8      <1>      cmp     al, bl ; process number comparison
779      0000CBFA 7413      <1>      je      short sysexit_10
780      0000CBFC 20C0      <1>      and     al, al
781      0000CBFE 7406      <1>      jz      short sysexit_9
782      <1>      sysexit_8:
783      0000CC00 FEC9      <1>      dec     cl
784      0000CC02 7416      <1>      jz      short sysexit_11
785      <1>      ; 08/01/2026 (BugFix)
786      0000CC04 B000      <1>      mov     al, 0
787      <1>      sysexit_9:
788      0000CC06 FECC      <1>      dec     ah
789      0000CC08 7412      <1>      jz      short sysexit_12
790      0000CC0A 83C610      <1>      add     esi, 16
791      0000CC0D EBE7      <1>      jmp     short sysexit_7
792      <1>
793      <1>      sysexit_10:
794      <1>      ; set timer event as free
795      0000CC0F C60600      <1>      mov     byte [esi], 0 ; 08/01/2026
796      <1>      ; mov     word [esi], 0
797      <1>      ; mov     dword [esi+12], 0
798      <1>      ;
799      0000CC12 FE0D[33830100] <1>      dec     byte [timer_events] ; 02/01/2017
800      <1>      ;
801      0000CC18 EBE6      <1>      jmp     short sysexit_8
802      <1>
803      <1>      sysexit_11:
804      <1>      ; sub     ax, ax ; 0 ; 26/02/2017
805      <1>      ; 30/07/2022
806      0000CC1A 29C0      <1>      sub     eax, eax ; 0
807      <1>      sysexit_12:
808      <1>      ; 26/02/2017 (Unlink IRQ callbacks belong to the user)
809      <1>      ; cmp     byte [u.irqc], 0 ; Count of IRQ callbacks
810      <1>      ; 08/01/2026
811      <1>      ; eax = 0
812      0000CC1C 3805[CE8E0100] <1>      cmp     byte [u.irqc], al ; 0 ?
813      0000CC22 7E53      <1>      jng     short sysexit_16 ; zero or invalid
814      <1>      ; 28/02/2017
815      <1>      ; clear IRQ callback flags (for 'sysrele' and 'sysret')
816      0000CC24 A2[CF8E0100] <1>      mov     [u.irqwait], al ; 0 ; force to clear waiting flag
817      0000CC29 A2[D08E0100] <1>      mov     [u.r_lock], al ; 0 ; force to clear busy flag
818      0000CC2E BE[80880100] <1>      mov     esi, IRQ.owner
819      <1>      sysexit_13:
820      <1>      ; lods     al, [u.uno] ; owner = current user ?
821      0000CC34 3A05[A58E0100] <1>      cmp     al, [u.uno] ; owner = current user ?
822      0000CC3A 7531      <1>      jne     short sysexit_14
823      0000CC3C C646FF00 <1>      mov     byte [esi-1], 0 ; owner = 0 : Free
824      <1>
825      <1>      ;;;
826      <1>      ; 22/08/2024 - Reset Hardware Interrupt Handler
827      0000CC40 89F1      <1>      mov     ecx, esi
828      0000CC42 81E9[80880100] <1>      sub     ecx, IRQ.owner
829      <1>      ; CL = (user configurable) IRQ index + 1 (1 to 9)
830      0000CC48 B8[A0350100] <1>      mov     eax, IRQenum
831      <1>      sysexit_18:
832      0000CC4D 3A08      <1>      cmp     cl, [eax]
833      0000CC4F 740A      <1>      je      short sysexit_19
834      0000CC51 3D[AF350100] <1>      cmp     eax, IRQenum+15
835      0000CC56 730D      <1>      jnb     short sysexit_20
836      0000CC58 40      <1>      inc     eax
837      0000CC59 EBF2      <1>      jmp     short sysexit_18

```

```

838
839 0000CC5B 2D[A0350100]
840
841
842
843 0000CC60 E8A7510000
844
845
846
847 0000CC65 FE0D[CE8E0100]
848 0000CC6B 7408
849
850 0000CC6D 81FE[88880100]
851 0000CC73 76BE
852
853
854
855 0000CC75 31C0
856
857 0000CC77 FB
858
859
860
861
862
863
864 0000CC78 E8CE350000
865
866
867
868
869
870
871 0000CC7D 40
872
873
874 0000CC7E 3C0A
875
876 0000CC80 72F6
877
878
879
880
881 0000CC82 BE[80880100]
882
883
884 0000CC87 8A1D[A58E0100]
885
886 0000CC8D 88A3[878D0100]
887
888
889 0000CC93 381D[FD880100]
890 0000CC99 7518
891
892 0000CC9B 883D[FD880100]
893
894 0000CCA1 8B0D[E4880100]
895 0000CCA7 09C9
896 0000CCA9 7408
897
898
899
900
901
902
903 0000CCAB 29C9
904 0000CCAD 890D[E4880100]
905
906
907
908 0000CCB3 D0E3
909
910
911
912 0000CCB5 668B8B[268D0100]
913
914 0000CCBC 668B93[468D0100]
915
916
917 0000CCC3 30DB
918
919 0000CCC5 31F6
920
921
922
923
924
925 0000CCC7 80C302
926
927
928 0000CCCA 66398B[468D0100]
929
930 0000CCD1 7513
931
932
933 0000CCD3 D0EB
934
935 0000CCD5 80BB[878D0100]03
936
937
938 0000CCDC 7506
939
940 0000CCDE 88A3[878D0100]
941
942
943
944 0000CCE4 D0E3
945
946
947
948
949 0000CCE6 663993[268D0100]
950
951 0000CCED 7502
952
953 0000CCEF 89DE
954
955
956
957
958 0000CCF1 80FB20
959
960 0000CCF4 72D1
961

```

```

<1> sysexit_19:
<1>     sub     eax, IRQenum
<1>     ; AL = IRQ number (will be reset to system's IRQ handler)
<1>     ; AH = 0
<1>     ;mov     ah, 0 ; reset
<1>     call    set_hardware_int_vector
<1> sysexit_20:
<1>     ;;;
<1>
<1>
<1>     dec     byte [u.irqc]
<1>     jz      short sysexit_15
<1> sysexit_14:
<1>     cmp     esi, IRQ.owner + 8 ; the last IRQ index number ?
<1>     jna     short sysexit_13 ; no
<1> sysexit_15:
<1>     ;xor     al, al ; 0
<1>     ; 22/08/2024
<1>     xor     eax, eax
<1> sysexit_16: ; 2:
<1>     sti     ; enable interrupts
<1>     ;
<1>     ; AX = 0
<1> sysexit_1: ; 1:
<1>     ; AX = File descriptor
<1>     ; / r1 has file descriptor (index to u.fp list)
<1>     ; / Search the whole list
<1>     call    fclose
<1>     ; jsr r0, fclose / close all files the process opened
<1>     ; ignore error return
<1>     ; br .+2 / ignore error return
<1>     ; inc ax
<1>     ; inc al
<1>     ; 22/08/2024
<1>     inc     eax
<1>     ; inc r1 / increment file descriptor
<1>     cmp     ax, 10
<1>     cmp     al, 10
<1>     ; cmp r1,$10. / end of u.fp list?
<1>     jb      short sysexit_1
<1>     ; blt 1b / no, go back
<1>
<1>     ; 22/08/2024
<1>     ; reset IRQs if owned by the current (dying) process
<1>     mov     esi, IRQ.owner
<1>
<1>
<1>     movzx   ebx, byte [u.uno]
<1>     mov     bl, [u.uno] ; 02/01/2017
<1>     ; movb u.uno,r1 / yes, move dying process's number to r1
<1>     mov     [ebx+p.stat-1], ah ; 0, SFREE
<1>     ; clrb p.stat-1(r1) / free the process
<1>     ; 10/04/2017
<1>     cmp     [audio_user], bl
<1>     jne     short sysexit_17
<1>     ; reset audio device (current) owner and 'initialized' flag
<1>     mov     [audio_user], bh ; 0
<1>     ; 27/05/2017
<1>     mov     ecx, [audio_buffer]
<1>     or      ecx, ecx
<1>     jz      short sysexit_17
<1>     ; 'deallocate_user_pages' is not necessary in sysexit !!!
<1>     ; push ebx
<1>     ; mov ebx, ecx
<1>     ; mov ecx, [audio_buff_size]
<1>     ; call deallocate_user_pages
<1>     ; (Modified Registers -> EAX, EDX, ESI, EDI, EBX, ECX, EBP)
<1>     sub     ecx, ecx
<1>     mov     [audio_buffer], ecx ; 0
<1>     pop     ebx
<1> sysexit_17:
<1>     ; shl bx, 1
<1>     shl     bl, 1
<1>     ; 22/08/2024
<1>     ; shl ebx, 1
<1>     ; asl r1 / use r1 for index into the below tables
<1>     mov     cx, [ebx+p.pid-2]
<1>     ; mov p.pid-2(r1),r3 / move dying process's name to r3
<1>     mov     dx, [ebx+p.ppid-2]
<1>     ; mov p.ppid-2(r1),r4 / move its parents name to r4
<1>     ; xor bx, bx ; 0
<1>     xor     bl, bl ; 0
<1>     ; clr r2
<1>     xor     esi, esi ; 0
<1>     ; clr r5 / initialize reg
<1> sysexit_2: ; 1:
<1>     ; / find children of this dying process,
<1>     ; / if they are zombies, free them
<1>     ; add bx, 2
<1>     add     bl, 2
<1>     ; add $2,r2 / search parent process table
<1>     ; / for dying process's name
<1>     cmp     [ebx+p.ppid-2], cx
<1>     ; cmp p.ppid-2(r2),r3 / found it?
<1>     jne     short sysexit_4
<1>     ; bne 3f / no
<1>     ; shr bx, 1
<1>     shr     bl, 1
<1>     ; asr r2 / yes, it is a parent
<1>     cmp     byte [ebx+p.stat-1], 3 ; SZOMB
<1>     ; cmpb p.stat-1(r2),$3 / is the child of this
<1>     ; / dying process a zombie
<1>     jne     short sysexit_3
<1>     ; bne 2f / no
<1>     mov     [ebx+p.stat-1], ah ; 0, SFREE
<1>     ; clrb p.stat-1(r2) / yes, free the child process
<1> sysexit_3: ; 2:
<1>     ; shr bx, 1
<1>     shl     bl, 1
<1>     ; asl r2
<1> sysexit_4: ; 3:
<1>     ; / search the process name table
<1>     ; / for the dying process's parent
<1>     cmp     [ebx+p.pid-2], dx
<1>     ; cmp p.pid-2(r2),r4 / found it?
<1>     jne     short sysexit_5
<1>     ; bne 3f / no
<1>     mov     esi, ebx
<1>     ; mov r2,r5 / yes, put index to p.pid table (parents
<1>     ; / process # x2) in r5
<1> sysexit_5: ; 3:
<1>     ; cmp bx, nproc + nproc
<1>     cmp     bl, nproc + nproc
<1>     ; cmp r2,$nproc+nproc / has whole table been searched?
<1>     jb      short sysexit_2
<1>     ; blt 1b / no, go back

```

```

962          <1>          ; mov r5,r1 / yes, r1 now has parents process # x2
963 0000CCF6 21F6      <1>          and     esi, esi ; r5=r1
964 0000CCF8 743F      <1>          jz      short sysexit_6
965          <1>          ; beq 2f / no parent has been found.
966          <1>          ; / The process just dies
967          <1>          ;shr     si, 1
968          <1>          ; 23/07/2022
969 0000CCFA D1EE      <1>          shr     esi, 1
970          <1>          ; asr r1 / set up index to p.stat
971 0000CCFC 8A86[878D0100] <1>          mov     al, [esi+p.stat-1]
972          <1>          ; movb p.stat-1(r1),r2 / move status of parent to r2
973 0000CD02 20C0      <1>          and     al, al
974 0000CD04 7433      <1>          jz      short sysexit_6
975          <1>          ; beq 2f / if its been freed, 2f
976 0000CD06 3C03      <1>          cmp     al, 3
977          <1>          ; cmp r2,$3 / is parent a zombie?
978 0000CD08 742F      <1>          je      short sysexit_6
979          <1>          ; beq 2f / yes, 2f
980          <1>
981          <1>          ; BH = 0
982 0000CD0A 8A1D[A58E0100] <1>          mov     bl, [u.uno]
983          <1>          ; movb u.uno,r3 / move dying process's number to r3
984 0000CD10 C683[878D0100]03 <1>          mov     byte [ebx+p.stat-1], 3 ; SZOMB
985          <1>          ; movb $3,p.stat-1(r3) / make the process a zombie
986          <1>
987 0000CD17 3C01      <1>          cmp     al, 1 ; SRUN
988 0000CD19 741E      <1>          je      short sysexit_6
989          <1>          ;cmp     al, 2
990          <1>          ; cmp r2,$2 / is the parent waiting for
991          <1>          ; / this child to die
992          <1>          ;jne     short sysexit_6
993          <1>          ; bne 2f / yes, notify parent not to wait any more
994          <1>          ; p.stat = 2 --> waiting
995          <1>          ; p.stat = 4 --> sleeping
996 0000CD1B C686[878D0100]01 <1>          mov     byte [esi+p.stat-1], 1 ; SRUN
997          <1>          ;;;
998          <1>          ; 18/08/2024 - exit code
999 0000CD22 A0[D28E0100] <1>          mov     al, [u.exit] ; exit code of the child
1000          <1>          ; 20/08/2024
1001 0000CD27 8883[278E0100] <1>          mov     [ebx+p.exittc-1], al ; save it to use by the parent
1002          <1>          ;;;
1003          <1>          ;dec     byte [esi+p.stat-1]
1004          <1>          ; decb p.stat-1(r1) / awaken it by putting it (parent)
1005          <1>          ;;;
1006          <1>          ;mov     ax, si ; r1 (process number in AL)
1007          <1>          ; 18/08/2024
1008 0000CD2D 89F0      <1>          mov     eax, esi
1009          <1>          ;;;
1010          <1>          ;mov     ebx, runq + 4
1011          <1>          ; mov $runq+4,r2 / on the runq
1012 0000CD2F BB[448E0100] <1>          mov     ebx, runq+2 ; normal run queue ; 02/01/2017
1013 0000CD34 E8FE3C0000 <1>          call    putlu
1014          <1>          ; jsr r0, putlu
1015          <1>          sysexit_6:
1016          <1>          ; / the process dies
1017 0000CD39 C605[A58E0100]00 <1>          mov     byte [u.uno], 0
1018          <1>          ; clrb u.uno / put zero as the process number,
1019          <1>          ; / so "swap" will
1020 0000CD40 E8F53B0000 <1>          call    swap
1021          <1>          ; jsr r0,swap / overwrite process with another process
1022          <1>          hlt_sys:
1023          <1>          ;sti
1024          <1>          hlts0:
1025 0000CD45 F4          <1>          hlt
1026 0000CD46 EBFD      <1>          jmp     short hlts0
1027          <1>          ; 0 / and thereby kill it; halt?
1028          <1>
1029          <1>          syswait: ; < wait for a process to die >
1030          <1>          ; 20/08/2024
1031          <1>          ; 30/07/2022
1032          <1>          ; 23/07/2022 - TRDOS 386 v2.0.5
1033          <1>          ; 17/09/2015
1034          <1>          ; 02/09/2015
1035          <1>          ; 01/09/2015
1036          <1>          ; 16/04/2015 (Retro UNIX 386 v1 - Beginning)
1037          <1>          ; 24/05/2013 - 05/02/2014 (Retro UNIX 8086 v1)
1038          <1>
1039          <1>          ; 'syswait' waits for a process die.
1040          <1>          ; It works in following way:
1041          <1>          ; 1) From the parent process number, the parent's
1042          <1>          ; process name is found. The p.ppid table of parent
1043          <1>          ; names is then searched for this process name.
1044          <1>          ; If a match occurs, r2 contains child's process
1045          <1>          ; number. The child status is checked to see if it is
1046          <1>          ; a zombie, i.e; dead but not waited for (p.stat=3)
1047          <1>          ; If it is, the child process is freed and it's name
1048          <1>          ; is put in (u.r0). A return is then made via 'sysret'.
1049          <1>          ; If the child is not a zombie, nothing happens and
1050          <1>          ; the search goes on through the p.ppid table until
1051          <1>          ; all processes are checked or a zombie is found.
1052          <1>          ; 2) If no zombies are found, a check is made to see if
1053          <1>          ; there are any children at all. If there are none,
1054          <1>          ; an error return is made. If there are, the parent's
1055          <1>          ; status is set to 2 (waiting for child to die),
1056          <1>          ; the parent is swapped out, and a branch to 'syswait'
1057          <1>          ; is made to wait on the next process.
1058          <1>
1059          <1>          ; Calling sequence:
1060          <1>          ; ?
1061          <1>          ; Arguments:
1062          <1>          ; -
1063          <1>          ; Inputs: -
1064          <1>          ; Outputs: if zombie found, it's name put in u.r0.
1065          <1>          ; .....
1066          <1>
1067          <1>
1068          <1>          ; / wait for a process to die
1069          <1>
1070          <1>          syswait_0:
1071 0000CD48 0FB61D[A58E0100] <1>          movzx    ebx, byte [u.uno] ; 01/09/2015
1072          <1>          ; movb u.uno,r1 / put parents process number in r1
1073 0000CD4F D0E3      <1>          shl     bl, 1
1074          <1>          ;shl     bx, 1
1075          <1>          ; asl r1 / x2 to get index into p.pid table
1076 0000CD51 668B83[268D0100] <1>          mov     ax, [ebx+p.pid-2]
1077          <1>          ; mov p.pid-2(r1),r1 / get the name of this process
1078 0000CD58 31F6      <1>          xor     esi, esi
1079          <1>          ; clr r2
1080 0000CD5A 31C9      <1>          xor     ecx, ecx ; 30/10/2013
1081          <1>          ;xor     cl, cl
1082          <1>          ; clr r3 / initialize reg 3
1083          <1>          syswait_1: ; 1:
1084          <1>          ;add     si, 2
1085          <1>          ; 23/07/2022

```



```

1086 0000CD5C 46      <1>      inc     esi
1087 0000CD5D 46      <1>      inc     esi
1088                <1>      ; add $2,r2 / use r2 for index into p.ppid table
1089                <1>      ; / search table of parent processes
1090                <1>      ; / for this process name
1091 0000CD5E 663B86[468D0100] <1>      cmp     ax, [esi+p.ppid-2]
1092                <1>      ; cmp p.ppid-2(r2),r1 / r2 will contain the childs
1093                <1>      ; / process number
1094 0000CD65 7542      <1>      jne     short syswait_3
1095                <1>      ; bne 3f / branch if no match of parent process name
1096                <1>      ; inc cx
1097 0000CD67 FEC1      <1>      inc     cl
1098                <1>      ; inc r3 / yes, a match, r3 indicates number of children
1099                <1>      ; shr si, 1
1100                <1>      ; 23/07/2022
1101 0000CD69 D1EE      <1>      shr     esi, 1
1102                <1>      ; asr r2 / r2/2 to get index to p.stat table
1103                <1>      ; The possible states ('p.stat' values) of a process are:
1104                <1>      ; 0 = free or unused
1105                <1>      ; 1 = active
1106                <1>      ; 2 = waiting for a child process to die
1107                <1>      ; 3 = terminated, but not yet waited for (zombie).
1108 0000CD6B 80BE[878D0100]03 <1>      cmp     byte [esi+p.stat-1], 3 ; SZOMB, 05/02/2014
1109                <1>      ; cmpb p.stat-1(r2),$3 / is the child process a zombie?
1110 0000CD72 7533      <1>      jne     short syswait_2
1111                <1>      ; bne 2f / no, skip it
1112 0000CD74 88BE[878D0100] <1>      mov     [esi+p.stat-1], bh ; 0
1113                <1>      ; clrb p.stat-1(r2) / yes, free it
1114                <1>
1115                <1>      ; 20/08/2024 (ebx = child's exit code)
1116 0000CD7A 31C0      <1>      xor     eax, eax
1117 0000CD7C 8A86[278E0100] <1>      mov     al, [esi+p.exittc-1] ; exit code
1118 0000CD82 8B2D[508E0100] <1>      mov     ebp, [u.usp]
1119                <1>      ; 22/08/2024
1120 0000CD88 894510      <1>      mov     [ebp+16], eax ; ebx
1121                <1>
1122                <1>      ; shl si, 1
1123                <1>      ; 23/07/2022
1124 0000CD8B D1E6      <1>      shl     esi, 1
1125                <1>      ; asl r2 / r2x2 to get index into p.pid table
1126                <1>      ; movzx eax, word [esi+p.pid-2]
1127                <1>      ; 20/08/2024
1128 0000CD8D 668B86[268D0100] <1>      mov     ax, [esi+p.pid-2]
1129 0000CD94 A3[548E0100] <1>      mov     [u.r0], eax
1130                <1>      ; mov p.pid-2(r2),*u.r0
1131                <1>      ; / put childs process name in (u.r0)
1132                <1>
1133                <1>      ; Retro UNIX 386 v1 modification ! (17/09/2015)
1134                <1>
1135                <1>      ; Parent process ID -p.ppid- field (of the child process)
1136                <1>      ; must be cleared in order to prevent infinitive 'syswait'
1137                <1>      ; system call loop from the application/program if it calls
1138                <1>      ; 'syswait' again (mistakenly) while there is not a zombie
1139                <1>      ; or running child process to wait. ('forktest.s', 17/09/2015)
1140                <1>
1141                <1>      ; Note: syswait will return with error if there is not a
1142                <1>      ; zombie or running process to wait.
1143                <1>
1144                <1>      ; sub ax, ax
1145                <1>      ; 30/07/2022
1146 0000CD99 29C0      <1>      sub     eax, eax ; 0
1147 0000CD9B 668986[468D0100] <1>      mov     [esi+p.ppid-2], ax ; 0 ; 17/09/2015
1148 0000CDA2 E9D9FCFFFF <1>      jmp     sysret0 ; ax = 0
1149                <1>
1150                <1>      ; jmp sysret
1151                <1>      ; br sysret1 / return cause child is dead
1152 0000CDA7 D1E6      <1>      syswait_2: ; 2:
1153                <1>      ; shl si, 1
1154                <1>      ; 23/07/2022
1155                <1>      ; shl esi, 1
1156                <1>      ; asl r2 / r2x2 to get index into p.ppid table
1157 0000CDA9 6683FE20 <1>      syswait_3: ; 3:
1158                <1>      cmp     si, nproc+nproc
1159                <1>      ; cmp r2,$nproc+nproc / have all processes been checked?
1160 0000CDAD 72AD      <1>      jb     short syswait_1
1161                <1>      ; blt 1b / no, continue search
1162                <1>      ; and cx, cx
1163 0000CDAF 20C9      <1>      and     cl, cl
1164                <1>      ; tst r3 / one gets here if there are no children
1165                <1>      ; / or children that are still active
1166                <1>      ; 30/10/2013
1167 0000CDB1 7515      <1>      jnz     short syswait_4
1168                <1>      ; jz error
1169                <1>      ; beq error1 / there are no children, error
1170 0000CDB3 890D[548E0100] <1>      mov     [u.r0], ecx ; 0
1171                <1>
1172                <1>      ; 20/08/2024 (ebx = child's exit code)
1173 0000CDB9 8B2D[508E0100] <1>      mov     ebp, [u.usp]
1174 0000CDBF 49        <1>      dec     ecx ; -1 ; 0FFFFFFFh
1175                <1>      ; 22/08/2024
1176 0000CDC0 894D10      <1>      mov     [ebp+16], ecx ; ebx
1177                <1>
1178 0000CDC3 E996FCFFFF <1>      jmp     error
1179                <1>      syswait_4:
1180 0000CDC8 8A1D[A58E0100] <1>      mov     bl, [u.uno]
1181                <1>      ; movb u.uno,r1 / there are children so put
1182                <1>      ; / parent process number in r1
1183 0000CDCE FE83[878D0100] <1>      inc     byte [ebx+p.stat-1] ; 2, SWAIT, 05/02/2014
1184                <1>      ; incb p.stat-1(r1) / it is waiting for
1185                <1>      ; / other children to die
1186                <1>      ; 04/11/2013
1187 0000CDD4 E8613B0000 <1>      call    swap
1188                <1>      ; jsr r0,swap / swap it out, because it's waiting
1189                <1>
1190                <1>      ; 23/08/2024 - bugfix
1191                <1>      ; restore [u.usp]
1192                <1>      ; (swap above changes [u.usp] just before wswap)
1193 0000CDD9 8925[508E0100] <1>      mov     [u.usp], esp
1194                <1>      ; points to user's regs on top ofn system stack
1195                <1>
1196 0000CDDF E964FFFFFF <1>      jmp     syswait_0
1197                <1>      ; br syswait / wait on next process
1198                <1>
1199                <1>      sysfork: ; < create a new process >
1200                <1>      ; 19/08/2024 - TRDOS 386 Kernel v2.0.9 (STDIN/STDOUT)
1201                <1>      ; 23/07/2022 - TRDOS 386 Kernel v2.0.5
1202                <1>      ; 02/01/2017 (TRDOS 386 modification)
1203                <1>      ; 04/09/2015 - 18/05/2015
1204                <1>      ; 28/08/2015 - 01/09/2015 - 02/09/2015
1205                <1>      ; 09/05/2015 - 10/05/2015 - 14/05/2015
1206                <1>      ; 06/05/2015 (Retro UNIX 386 v1 - Beginning)
1207                <1>      ; 24/05/2013 - 14/02/2014 (Retro UNIX 8086 v1)
1208                <1>
1209                <1>      ; 'sysfork' creates a new process. This process is referred

```

```

1210 <1> ; to as the child process. This new process core image is
1211 <1> ; a copy of that of the caller of 'sysfork'. The only
1212 <1> ; distinction is the return location and the fact that (u.r0)
1213 <1> ; in the old process (parent) contains the process id (p.pid)
1214 <1> ; of the new process (child). This id is used by 'syswait'.
1215 <1> ; 'sysfork' works in the following manner:
1216 <1> ; 1) The process status table (p.stat) is searched to find
1217 <1> ; a process number that is unused. If none are found
1218 <1> ; an error occurs.
1219 <1> ; 2) when one is found, it becomes the child process number
1220 <1> ; and it's status (p.stat) is set to active.
1221 <1> ; 3) If the parent had a control tty, the interrupt
1222 <1> ; character in that tty buffer is cleared.
1223 <1> ; 4) The child process is put on the lowest priority run
1224 <1> ; queue via 'putlu'.
1225 <1> ; 5) A new process name is gotten from 'mpid' (actually
1226 <1> ; it is a unique number) and is put in the child's unique
1227 <1> ; identifier; process id (p.pid).
1228 <1> ; 6) The process name of the parent is then obtained and
1229 <1> ; placed in the unique identifier of the parent process
1230 <1> ; name is then put in 'u.r0'.
1231 <1> ; 7) The child process is then written out on disk by
1232 <1> ; 'wswap', i.e., the parent process is copied onto disk
1233 <1> ; and the child is born. (The child process is written
1234 <1> ; out on disk/drum with 'u.uno' being the child process
1235 <1> ; number.)
1236 <1> ; 8) The parent process number is then restored to 'u.uno'.
1237 <1> ; 9) The child process name is put in 'u.r0'.
1238 <1> ; 10) The pc on the stack sp + 18 is incremented by 2 to
1239 <1> ; create the return address for the parent process.
1240 <1> ; 11) The 'u.fp' list is then searched to see what files
1241 <1> ; the parent has opened. For each file the parent has
1242 <1> ; opened, the corresponding 'fsp' entry must be updated
1243 <1> ; to indicate that the child process also has opened
1244 <1> ; the file. A branch to 'sysret' is then made.
1245 <1>
1246 <1> Calling sequence:
1247 <1> from shell ?
1248 <1> Arguments:
1249 <1> -
1250 <1> Inputs: -
1251 <1> Outputs: *u.r0 - child process name
1252 <1> .....
1253 <1>
1254 <1> Retro UNIX 8086 v1 modification:
1255 <1> AX = r0 = PID (>0) (at the return of 'sysfork')
1256 <1> = process id of child a parent process returns
1257 <1> = process id of parent when a child process returns
1258 <1>
1259 <1> In original UNIX v1, sysfork is called and returns as
1260 <1> in following manner: (with an example: c library, fork)
1261 <1>
1262 <1> 1:
1263 <1> sys fork
1264 <1> br 1f / child process returns here
1265 <1> bes 2f / parent process returns here
1266 <1> / pid of new process in r0
1267 <1> rts pc
1268 <1> 2: / parent process conditionally branches here
1269 <1> mov $-1,r0 / pid = -1 means error return
1270 <1> rts pc
1271 <1>
1272 <1> 1: / child process branches here
1273 <1> clr r0 / pid = 0 in child process
1274 <1> rts pc
1275 <1>
1276 <1> In UNIX v7x86 (386) by Robert Nordier (1999)
1277 <1> // pid = fork();
1278 <1> //
1279 <1> // pid == 0 in child process;
1280 <1> // pid == -1 means error return
1281 <1> // in child,
1282 <1> // parents id is in par_uid if needed
1283 <1>
1284 <1> _fork:
1285 <1> mov $.fork,eax
1286 <1> int $0x30
1287 <1> jmp 1f
1288 <1> jnc 2f
1289 <1> jmp cerror
1290 <1>
1291 <1> 1: mov eax,_par_uid
1292 <1> xor eax,eax
1293 <1>
1294 <1> 2: ret
1295 <1>
1296 <1> In Retro UNIX 8086 v1,
1297 <1> 'sysfork' returns in following manner:
1298 <1>
1299 <1> mov ax, sys_fork
1300 <1> mov bx, offset @f ; routine for child
1301 <1> int 20h
1302 <1> jc error
1303 <1>
1304 <1> ; Routine for parent process here (just after 'jc')
1305 <1> mov word ptr [pid_of_child], ax
1306 <1> jmp next_routine_for_parent
1307 <1>
1308 <1> @@: ; routine for child process here
1309 <1> ....
1310 <1> NOTE: 'sysfork' returns to specified offset
1311 <1> for child process by using BX input.
1312 <1> (at first, parent process will return then
1313 <1> child process will return -after swapped in-
1314 <1> 'syswait' is needed in parent process
1315 <1> if return from child process will be waited for.)
1316 <1>
1317 <1>
1318 <1> ; / create a new process
1319 <1> ; EBX = return address for child process
1320 <1> ; (Retro UNIX 8086 v1 modification !)
1321 <1> xor esi, esi
1322 <1> ; clr r1
1323 <1> sysfork_1: ; 1: / search p.stat table for unused process number
1324 <1> inc esi
1325 <1> ; inc r1
1326 <1> cmp byte [esi+p.stat-1], 0 ; SFREE, 05/02/2014
1327 <1> ; tstb p.stat-1(r1) / is process active, unused, dead
1328 <1> jna short sysfork_2
1329 <1> ; beq 1f / it's unused so branch
1330 <1> si, nproc
1331 <1> ; cmp r1,$nproc / all processes checked
1332 <1> short sysfork_1
1333 <1> ; blt 1b / no, branch back

```

```

1334      <1>      ;
1335      <1>      ; Retro UNIX 8086 v1. modification:
1336      <1>      ; Parent process returns from 'sysfork' to address
1337      <1>      ; which is just after 'sysfork' system call in parent
1338      <1>      ; process. Child process returns to address which is put
1339      <1>      ; in BX register by parent process for 'sysfork'.
1340      <1>      ;
1341      <1>      ; add $2,18.(sp) / add 2 to pc when trap occurred, points
1342      <1>      ; / to old process return
1343      <1>      ; br error1 / no room for a new process
1344      <1>      sysfork_err:
1345      <1>      jmp     error
1346      <1>      sysfork_2: ; 1:
1347      <1>      call    allocate_page
1348      <1>      ;jc     error
1349      <1>      ; 23/07/2022
1350      <1>      jc      short sysfork_err
1351      <1>      push    eax ; UPAGE (user structure page) address
1352      <1>      ; Retro UNIX 386 v1 modification!
1353      <1>      call    duplicate_page_dir
1354      <1>      ; EAX = New page directory
1355      <1>      jnc     short sysfork_3
1356      <1>      pop     eax ; UPAGE (user structure page) address
1357      <1>      call    deallocate_page
1358      <1>      jmp     error
1359      <1>      sysfork_3:
1360      <1>      ; Retro UNIX 386 v1 modification !
1361      <1>      push    esi
1362      <1>      call    wswap ; save current user (u) structure, user registers
1363      <1>      ; and interrupt return components (for IRET)
1364      <1>      xchg    eax, [u.pgdir] ; page directory of the child process
1365      <1>      mov     [u.ppgdir], eax ; page directory of the parent process
1366      <1>      pop     esi
1367      <1>      pop     eax ; UPAGE (user structure page) address
1368      <1>      ; [u.usp] = esp
1369      <1>      mov     edi, esi
1370      <1>      shl     di, 2
1371      <1>      ; 23/07/2022
1372      <1>      shl     edi, 2
1373      <1>      mov     [edi+p.upage-4], eax ; memory page for 'user' struct
1374      <1>      mov     [u.upage], eax ; memory page for 'user' struct (child)
1375      <1>      ;;;
1376      <1>      ; 19/08/2024 - reset STDIN/STDOUT redirections (and ungetchar)
1377      <1>      xor     eax, eax ; 0
1378      <1>      ; 19/08/2024
1379      <1>      ;mov    [u.stdin], al ; 0
1380      <1>      ;mov    [u.stdout], al ; 0
1381      <1>      ;mov    [u.ungetc], al ; 0
1382      <1>      ;mov    [u.getc], al ; 0
1383      <1>      mov     [u.stdin], eax ; 0
1384      <1>      ;;;
1385      <1>      ;
1386      <1>      ; 28/08/2015
1387      <1>      ;movzx  eax, byte [u.uno] ; parent process number
1388      <1>      ; 19/08/2024
1389      <1>      ; eax = 0
1390      <1>      mov     al, [u.uno] ; parent process number
1391      <1>      ; movb u.uno,-(sp) / save parent process number
1392      <1>      mov     edi, eax
1393      <1>      push    eax ; **
1394      <1>      mov     al, [edi+p.ttyc-1] ; console tty (parent)
1395      <1>      ; 18/09/2015
1396      <1>      ;mov    [esi+p.ttyc-1], al ; set child's console tty
1397      <1>      ;mov    [esi+p.waitc-1], ah ; 0 ; reset child's wait channel
1398      <1>      ;mov    [esi+p.ttyc-1], ax ; al - set child's console tty
1399      <1>      ; ah - reset child's wait channel
1400      <1>      ; 23/07/2022
1401      <1>      mov     [esi+p.ttyc-1], al ; set child's console tty
1402      <1>      mov     eax, esi
1403      <1>      mov     [u.uno], al ; child process number
1404      <1>      ;movb r1,u.uno / set child process number to r1
1405      <1>      inc     byte [esi+p.stat-1] ; 1, SRUN, 05/02/2014
1406      <1>      ; incb p.stat-1(r1) / set p.stat entry for child
1407      <1>      ; / process to active status
1408      <1>      ; mov u.ttyp,r2 / put pointer to parent process'
1409      <1>      ; / control tty buffer in r2
1410      <1>      ; beq 2f / branch, if no such tty assigned
1411      <1>      ; clrb 6(r2) / clear interrupt character in tty buffer
1412      <1>      ; 2:
1413      <1>      push    ebx ; * return address for the child process
1414      <1>      ; * Retro UNIX 8086 v1 feature only !
1415      <1>      ; (Retro UNIX 8086 v1 modification!)
1416      <1>      ; mov $runq+4,r2
1417      <1>      mov     ebx, runq+2 ; normal run queue ; 02/01/2017
1418      <1>      call    putlu
1419      <1>      ; jsr r0,putlu / put child process on lowest priority
1420      <1>      ; / run queue
1421      <1>      ; 23/07/2022
1422      <1>      shl     esi, 1
1423      <1>      ;shl    si, 1
1424      <1>      ; asl r1 / multiply r1 by 2 to get index
1425      <1>      ; / into p.pid table
1426      <1>      inc     word [mpid]
1427      <1>      ; inc mpid / increment m.pid; get a new process name
1428      <1>      mov     ax, [mpid]
1429      <1>      mov     [esi+p.pid-2], ax
1430      <1>      ;mov mpid,p.pid-2(r1) / put new process name
1431      <1>      ; / in child process' name slot
1432      <1>      pop     edx ; * return address for the child process
1433      <1>      ; * Retro UNIX 8086 v1 feature only !
1434      <1>      pop     ebx ; **
1435      <1>      ;mov    [esp], ** parent process number
1436      <1>      ; movb (sp),r2 / put parent process number in r2
1437      <1>      ; 23/07/2022
1438      <1>      shl     ebx, 1
1439      <1>      ;shl    bx, 1
1440      <1>      ;asl r2 / multiply by 2 to get index into below tables
1441      <1>      ;movzx  eax, word [ebx+p.pid-2]
1442      <1>      mov     ax, [ebx+p.pid-2]
1443      <1>      ; mov p.pid-2(r2),r2 / get process name of parent
1444      <1>      ; / process
1445      <1>      mov     [esi+p.ppid-2], ax
1446      <1>      ; mov r2,p.ppid-2(r1) / put parent process name
1447      <1>      ; / in parent process slot for child
1448      <1>      mov     [u.r0], eax
1449      <1>      ; mov r2,*u.r0 / put parent process name on stack
1450      <1>      ; / at location where r0 was saved
1451      <1>      mov     ebp, [u.sp] ; points to return address (EIP for IRET)
1452      <1>      mov     [ebp], edx ; *, CS:EIP -> EIP
1453      <1>      ; * return address for the child process
1454      <1>      ; mov $sysret1,-(sp) /
1455      <1>      ; mov sp,u.usp / contents of sp at the time when
1456      <1>      ; / user is swapped out
1457      <1>      ; mov $sstack,sp / point sp to swapping stack space

```

```

1458 <1> ; 04/09/2015 - 01/09/2015
1459 <1> ; [u.usp] = esp
1460 0000CEA1 68[7ECA0000] <1> push sysret ; ***
1461 0000CEA6 8925[508E0100] <1> mov [u.usp], esp ; points to 'sysret' address (***)
1462 <1> ; (for child process)
1463 0000CEAC 31C0 <1> xor eax, eax
1464 0000CEAE 66A3[888E0100] <1> mov [u.ttyp], ax ; 0
1465 <1> ;
1466 0000CEB4 E80E3B0000 <1> call wswap ; Retro UNIX 8086 v1 modification !
1467 <1> ; jsr r0,wswap / put child process out on drum
1468 <1> ; jsr r0,unpack / unpack user stack
1469 <1> ; mov u.usp,sp / restore user stack pointer
1470 <1> ; tst (sp)+ / bump stack pointer
1471 <1> ; Retro UNIX 386 v1 modification !
1472 0000CEB9 58 <1> pop eax ; ***
1473 <1> ; shl bx, 1
1474 <1> ; 23/07/2022
1475 0000CEBA D1E3 <1> shl ebx, 1
1476 0000CEBC 8B83[948D0100] <1> mov eax, [ebx+p.upage-4] ; UPAGE address ; 14/05/2015
1477 0000CEC2 E8383B0000 <1> call rswap ; restore parent process 'u' structure,
1478 <1> ; registers and return address (for IRET)
1479 <1> ; movb (sp)+,u.uno / put parent process number in u.uno
1480 0000CEC7 0FB705[3C8E0100] <1> movzx eax, word [mpid]
1481 0000CECE A3[548E0100] <1> mov [u.r0], eax
1482 <1> ; mov mpid,*u.r0 / put child process name on stack
1483 <1> ; / where r0 was saved
1484 <1> ; add $2,18.(sp) / add 2 to pc on stack; gives parent
1485 <1> ; / process return
1486 <1> ; xor ebx, ebx
1487 0000CED3 31F6 <1> xor esi, esi
1488 <1> ; clr r1
1489 <1> sysfork_4: ; 1: / search u.fp list to find the files
1490 <1> ; / opened by the parent process
1491 <1> ; 01/09/2015
1492 <1> ; xor bh, bh
1493 <1> ; mov bl, [esi+u.fp]
1494 0000CED5 8A86[5E8E0100] <1> mov al, [esi+u.fp]
1495 <1> ; movb u.fp(r1),r2 / get an open file for this process
1496 <1> ; or bl, bl
1497 0000CEDB 08C0 <1> or al, al
1498 0000CEDD 7406 <1> jz short sysfork_5
1499 <1> ; beq 2f / file has not been opened by parent,
1500 <1> ; / so branch
1501 <1> ; mov ah, 10 ; Retro UNIX 386 v1 fsp structure size = 10 bytes
1502 <1> ; mul ah
1503 <1> ; 23/07/2022
1504 <1> ; Retro UNIX 386 v2 & TRDOS 386 v2.0.5 fsp struc size = 16 bytes
1505 <1> ; mov ebx, eax
1506 <1> ; shl ebx, 4 ; * 16
1507 <1> ; inc byte [ebx+fsp-10]
1508 <1> ;
1509 <1> ; 23/07/2022 (BugFix)
1510 0000CEDF FE80[44840100] <1> inc byte [OF_OPENCOUNT+eax]
1511 <1> ;
1512 <1> ; movzx ebx, ax
1513 <1> ; mov bx, ax
1514 <1> ; shl bx, 3
1515 <1> ; asl r2 / multiply by 8
1516 <1> ; asl r2 / to get index into fsp table
1517 <1> ; asl r2
1518 <1> ; inc byte [ebx+fsp-2]
1519 <1> ; incb fsp-2(r2) / increment number of processes
1520 <1> ; / using file, because child will now be
1521 <1> ; / using this file
1522 <1> sysfork_5: ; 2:
1523 0000CEE5 46 <1> inc esi
1524 <1> ; inc r1 / get next open file
1525 <1> ; 23/07/2022
1526 0000CEE6 6683FE20 <1> cmp si, OPENFILES ; = 10
1527 <1> ; cmp si, 10
1528 <1> ; cmp r1,$10. / 10. files is the maximum number which
1529 <1> ; / can be opened
1530 0000CEEA 72E9 <1> jb short sysfork_4
1531 <1> ; blt 1b / check next entry
1532 0000CEEC E98DFBFFFF <1> jmp sysret
1533 <1> ; br sysret1
1534 <1> ;
1535 <1> syscreat: ; < create file >
1536 <1> ; 29/08/2024
1537 <1> ; 25/08/2024 - TRDOS 386 v2.0.9
1538 <1> ; 08/08/2022
1539 <1> ; 23/07/2022 - TRDOS 386 kernel v2.0.5
1540 <1> ; 13/11/2017
1541 <1> ; 27/10/2016
1542 <1> ; 25/10/2016 - 26/10/2016
1543 <1> ; 15/10/2016 - 16/10/2016 - 17/10/2016
1544 <1> ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)
1545 <1> ; -derived from INT_21h.ASM-
1546 <1> ; ("loc_INT21h_create_file")
1547 <1> ;
1548 <1> ; 10/07/2011 (12/03/2011)
1549 <1> ; INT 21h Function AH = 3Ch
1550 <1> ; Create File
1551 <1> ; INPUT
1552 <1> ; CX = Attributes
1553 <1> ; DS:DX= Address of zero terminated path name
1554 <1> ;
1555 <1> ; 27/12/2015 (Retro UNIX 386 v1.1)
1556 <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1557 <1> ; 27/05/2013 (Retro UNIX 8086 v1)
1558 <1> ;
1559 <1> ; 'syscreat' called with two arguments; name and mode.
1560 <1> ; u.namep points to name of the file and mode is put
1561 <1> ; on the stack. 'namei' is called to get i-number of the file.
1562 <1> ; If the file already exists, it's mode and owner remain
1563 <1> ; unchanged, but it is truncated to zero length. If the file
1564 <1> ; did not exist, an i-node is created with the new mode via
1565 <1> ; 'maknod' whether or not the file already existed, it is
1566 <1> ; open for writing. The fsp table is then searched for a free
1567 <1> ; entry. When a free entry is found, proper data is placed
1568 <1> ; in it and the number of this entry is put in the u.fp list.
1569 <1> ; The index to the u.fp (also know as the file descriptor)
1570 <1> ; is put in the user's r0.
1571 <1> ;
1572 <1> ; Calling sequence:
1573 <1> ; syscreate; name; mode
1574 <1> ; Arguments:
1575 <1> ; name - name of the file to be created
1576 <1> ; mode - mode of the file to be created
1577 <1> ; Inputs: (arguments)
1578 <1> ; Outputs: *u.r0 - index to u.fp list
1579 <1> ; (the file descriptor of new file)
1580 <1> ; .....
1581 <1> ; Retro UNIX 8086 v1 modification:

```

```

1582      ; 'syscreate' system call has two arguments; so,
1583      ; * 1st argument, name is pointed to by BX register
1584      ; * 2nd argument, mode is in CX register
1585      ;
1586      ; AX register (will be restored via 'u.r0') will return
1587      ; to the user with the file descriptor/number
1588      ; (index to u.fp list).
1589      ;
1590      ;call arg2
1591      ; * name - 'u.namep' points to address of file/path name
1592      ;         in the user's program segment ('u.segmt')
1593      ;         with offset in BX register (as sysopen argument 1).
1594      ; * mode - sysopen argument 2 is in CX register
1595      ;         which is on top of stack.
1596      ;
1597      ; TRDOS 386 (10/10/2016)
1598      ;
1599      ; INPUT ->
1600      ; CL = File Attributes
1601      ; bit 0 (1) - Read only file (R)
1602      ; bit 1 (1) - Hidden file (H)
1603      ; bit 2 (1) - System file (R)
1604      ; bit 3 (1) - Volume label/name (V)
1605      ; bit 4 (1) - Subdirectory (D)
1606      ; bit 5 (1) - File has been archived (A)
1607      ; EBX = Pointer to filename (ASCII) -path-
1608      ;
1609      ; OUTPUT ->
1610      ; eax = File/Device Handle/Number (index) (AL)
1611      ; cf = 1 -> Error code in AL
1612      ;
1613      ; Modified Registers: EAX (at the return of system call)
1614      ;
1615      ; Note: If the file is existing and it has not any one
1616      ; of S,H,R,V,D attributes, it will be truncated
1617      ; to zero length; otherwise, access error will be
1618      ; returned.
1619      ;
1620      ; sysmkdir_0:
1621      test    cl, 08h ; volume name
1622      jz      short syscreat_0
1623      ;
1624      ; Volume name or long name creation
1625      ; is not permitted (in TRDOS 386)!
1626      mov     eax, ERR_FILE_ACCESS ; 11 ; 'permission denied !'
1627      jmp     sysopen_dev_err ; 08/08/2022
1628      ; syscreat_0:
1629      ;mov     [u.namep], ebx
1630      push    ecx
1631      mov     esi, ebx
1632      ; file name is forced, change directory as temporary
1633      mov     ax, 1
1634      mov     [FFF_valid], ah ; 0 ; reset ; 17/10/2016
1635      call    set_working_path
1636      call    set_working_path_x ; 17/10/2016
1637      jc      short syscreat_err
1638      ; 23/07/2022
1639      jnc     short syscreat_3
1640      jmp     syscreat_err
1641      ;
1642      ; syscreat_3:
1643      ; 16/10/2016
1644      cmp     byte [SWP_inv_fname], 0
1645      ja      short syscreat_inv_fname ; invalid file name !
1646      ;
1647      ; Here, we have a valid path and also a valid file name
1648      ; (working dir has been changed if the path
1649      ; -file name string- had contained a dir name.)
1650      ;
1651      xor     ax, ax
1652      ; 25/08/2024
1653      xor     eax, eax
1654      mov     esi, FindFile_Name
1655      call    find_first_file
1656      pop     ecx
1657      ; ESI = Directory Entry (FindFile_DirEntry) Location
1658      ; EDI = Directory Buffer Directory Entry Location
1659      ; EAX = File Size
1660      ; BL = Attributes of The File/Directory
1661      ; BH = Long Name Yes/No Status (>0 is YES)
1662      ; DX > 0 : Ambiguous filename chars are used
1663      jc      short syscreat_1 ; file not found (the good!)
1664      ; or another error (the bad!)
1665      ;
1666      ; (& the ugly!) truncate file to zero length before open
1667      ;
1668      ; '*' and '?' already checked at 'set_working_path' stage
1669      and     dx, dx
1670      jnz     short sysmkdir_err ; permission denied
1671      ; invalid filename chars
1672      ;
1673      test    cl, 10h ; subdirectory ?
1674      jnz     short sysmkdir_err
1675      ;
1676      ; BL = File Attributes:
1677      ; bit 0 (1) - Read only file (R)
1678      ; bit 1 (1) - Hidden file (H)
1679      ; bit 2 (1) - System file (R)
1680      ; bit 3 (1) - Volume label/name (V)
1681      ; bit 4 (1) - Subdirectory (D)
1682      ; bit 5 (1) - File has been archived
1683      ;
1684      ; * existing directory must not be truncated
1685      ; (we don't know it is empty or not, at this stage)
1686      ; * existing volume name (or a long name) can not be
1687      ; re-created or truncated by 'syscreat'
1688      ; * A file with S, H, R attributes must not be truncated
1689      ; (change attributes to normal, if you need truncate it)
1690      ;
1691      test    bl, 00011111b ; check attributes of existing file
1692      jnz     short sysmkdir_err
1693      ;
1694      ; normal file, OK to continue...
1695      ;
1696      ; ESI = FindFile_DirEntry
1697      mov     ax, [esi+DirEntry_FstClusHI] ; 20
1698      shl     eax, 16 ; 13/11/2017
1699      mov     ax, [esi+DirEntry_FstClusLO] ; 26
1700      ;
1701      ; 24/08/2024
1702      and     eax, eax
1703      jz      short skip_truncate ; first cluster is 0 !
1704      ;
1705      ; EAX = First cluster to be truncated/unlinked

```

```

1706 0000CF36 57          <1>    push    edi
1707 0000CF37 51          <1>    push    ecx
1708 0000CF38 BE00010900    <1>    mov     esi, Logical_DOSDisks
1709 0000CF3D 29C9          <1>    sub     ecx, ecx
1710          <1>    ;mov    ch, [Current_Drv] ; = [FindFile_Drv]
1711          <1>    ; 03/09/2024
1712 0000CF3F 8A2D[02800100]    <1>    mov     ch, [FindFile_Drv]
1713 0000CF45 01CE          <1>    add     esi, ecx
1714          <1>    ; ESI = Logical dos drive description table address
1715 0000CF47 E8C2F7FFFF    <1>    call    truncate_cluster_chain
1716 0000CF4C 59          <1>    pop     ecx
1717 0000CF4D 5F          <1>    pop     edi
1718 0000CF4E 7237          <1>    jc      short syscreate_truncate_err
1719          <1>
1720          <1>    ; 03/09/2024
1721 0000CF50 66C747140000    <1>    mov     word [edi+DirEntry_FstClusHI], 0
1722 0000CF56 66C7471A0000    <1>    mov     word [edi+DirEntry_FstClusLO], 0
1723          <1>
1724          <1>    ; 24/08/2024
1725          <1>    skip_truncate:
1726          <1>    ; 26/10/2016
1727          <1>    ; EDI = Directory entry address in directory buffer
1728          <1>    ; Update directory entry
1729 0000CF5C E88BDDFFFF    <1>    call    convert_current_date_time
1730          <1>    ; OUTPUT -> DX = Date in dos dir entry format
1731          <1>    ;      AX = Time in dos dir entry format
1732 0000CF61 66894716    <1>    mov     [edi+DirEntry_wrtTime], ax
1733 0000CF65 66895718    <1>    mov     [edi+DirEntry_wrtDate], dx
1734 0000CF69 66895712    <1>    mov     [edi+DirEntry_LastAccDate], dx
1735 0000CF6D 31C0          <1>    xor     eax, eax ; file size = 0
1736 0000CF6F 89471C          <1>    mov     [edi+DirEntry_FileSize], eax ; 0
1737 0000CF72 C605[897E0100]02    <1>    mov     byte [DirBuff_ValidData], 2 ; data changed sign
1738          <1>    ;mov    esi, FindFile_DirEntry
1739          <1>    ; 03/09/2024
1740 0000CF79 B201          <1>    mov     dl, 1 ; open file for writing
1741 0000CF7B E9AB000000    <1>    jmp     sysopen_2 ; 08/08/2022
1742          <1>
1743          <1>    sysmkdir_err:
1744          <1>    ; 1 = write, 2 = read & write, >2 = invalid
1745 0000CF80 B80B000000    <1>    mov     eax, ERR_FILE_ACCESS ; 11 ; 'permission denied !'
1746 0000CF85 EB74          <1>    jmp     short sysopen_err
1747          <1>
1748          <1>    syscreate_truncate_err:
1749 0000CF87 B812000000    <1>    mov     eax, ERR_DRV_WRITE ; 18 ; 'disk write error !'
1750 0000CF8C EB6D          <1>    jmp     short sysopen_err
1751          <1>
1752          <1>    syscreat_inv_fname: ; invalid file name chars
1753          <1>    ; 16/10/2016
1754 0000CF8E B81A000000    <1>    mov     eax, ERR_INV_FILE_NAME ; 26 ; invalid file name chars
1755 0000CF93 59          <1>    pop     ecx
1756 0000CF94 EB65          <1>    jmp     short sysopen_err
1757          <1>
1758          <1>    syscreat_1:
1759          <1>    ; Error code in EAX
1760 0000CF96 3C02          <1>    cmp     al, 02h ; 'File not found' error
1761 0000CF98 7561          <1>    jne     short sysopen_err
1762          <1>
1763 0000CF9A F6C110          <1>    test    cl, 10h ; Directory
1764          <1>    ;jnz    sysmkdir_2
1765          <1>    ; 23/07/2022
1766 0000CF9D 7405          <1>    jz      short syscreat_2
1767 0000CF9F E9C3010000    <1>    jmp     sysmkdir_2
1768          <1>
1769          <1>    syscreat_2:
1770          <1>    mov     esi, FindFile_Name
1771          <1>    ;xor     edx, edx
1772          <1>    xor     eax, eax ; File Size = 0
1773 0000CFAB 31DB          <1>    xor     ebx, ebx
1774 0000CFAD 4B          <1>    dec     ebx ; FFFFFFFFh -> create empty file
1775          <1>    ;      (only for FAT fs)
1776          <1>    ; CL = File Attributes
1777 0000CFAE E8B0ECFFFF    <1>    call    create_file
1778 0000CFB3 7246          <1>    short sysopen_err
1779          <1>    ; EAX = New file's first cluster
1780          <1>    ; ESI = Logical Dos Dry Descr. Table Addr.
1781          <1>    ; EBX = offset CreateFile_Size
1782          <1>    ; ECX = Sectors per cluster (<256)
1783          <1>    ; EDX = Directory entry index/number (<65536)
1784          <1>    ; 29/08/2024
1785          <1>    ; EBX = File Size (0 for a new, empty file)
1786          <1>    ; ECX = Directory Entry Index/Number (<2048)
1787          <1>    ;      (in directory cluster, not in directory)
1788          <1>    ; EDX = Directory Cluster Number (of the file)
1789          <1>
1790          <1>    ; 26/10/2016
1791          <1>    ;mov    esi, Directory_Buffer
1792          <1>    ;shl    dx, 5 ; *32
1793          <1>    ;add    esi, edx
1794          <1>    ; esi = directory entry address in directory buffer
1795          <1>
1796          <1>    ; Here, directory entry has been created but last
1797          <1>    ; modification date & time of the parent dir has not
1798          <1>    ; been updated, yet!
1799          <1>    ; (Note: Directory and FAT buffers have been updated...)
1800          <1>
1801 0000CFB5 E850DEFFFF    <1>    call    update_parent_dir_lmdt ; now, it is OK too!
1802          <1>
1803          <1>    ; 25/10/2016
1804 0000CFBA 66B80018    <1>    mov     ax, 1800h
1805 0000CFBE BE[44800100]    <1>    mov     esi, FindFile_Name
1806 0000CFC3 E8A2B9FFFF    <1>    call    find_first_file
1807 0000CFC8 7231          <1>    jc      short sysopen_err
1808          <1>
1809          <1>    ; Only possible error after here is
1810          <1>    ; "too many open files !" error.
1811          <1>    ;
1812          <1>    ; If "syscreat" will return with that error,
1813          <1>    ; (the file has been created but it could not be opened)
1814          <1>    ; the user must retry to open this file again
1815          <1>    ; or must close another file before using
1816          <1>    ; "sysopen" system call.
1817          <1>
1818 0000CFCA B201          <1>    mov     dl, 1 ; open file for writing
1819          <1>    ; ESI = Directory Entry (FindFile_DirEntry) Location
1820          <1>    ; EAX = File Size (= 0)
1821 0000CFCC EB5D          <1>    jmp     short sysopen_2
1822          <1>
1823          <1>    sysopen: ;<open file>
1824          <1>    ; 19/12/2025 - TRDOS 386 v2.0.10
1825          <1>    ; 03/09/2024
1826          <1>    ; 19/08/2024 - TRDOS 386 v2.0.9
1827          <1>    ; 23/07/2022 - TRDOS 386 v2.0.5
1828          <1>    ; 17/04/2021 - TRDOS 386 v2.0.4
1829          <1>    ;      (temporary modifications)

```

```

1830      <1>      ; 26/10/2016
1831      <1>      ; 24/10/2016
1832      <1>      ; 17/10/2016
1833      <1>      ; 15/10/2016
1834      <1>      ; 06/10/2016, 07/10/2016, 08/10/2016
1835      <1>      ; 05/10/2016 (TRDOS 386 = TRDOS v2.0)
1836      <1>      ; -derived from INT_21H.ASM-
1837      <1>      ; ("loc_INT21h_open_file")
1838      <1>      ;
1839      <1>      ; 26/02/2011
1840      <1>      ; INT 21h Function AH = 3Dh
1841      <1>      ; Open File
1842      <1>      ; INPUT
1843      <1>      ; AL= File Access Value
1844      <1>      ; 0- Open for reading
1845      <1>      ; 1- Open for writing
1846      <1>      ; 2- Open for reading and writing
1847      <1>      ; DS:DX= Pointer to filename (ASCIIIZ)
1848      <1>      ;
1849      <1>      ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
1850      <1>      ; 22/05/2013 - 27/05/2013 (Retro UNIX 8086 v1)
1851      <1>      ; 'sysopen' opens a file in following manner:
1852      <1>      ; 1) The second argument in a sysopen says whether to
1853      <1>      ; open the file ro read (0) or write (>0).
1854      <1>      ; 2) I-node of the particular file is obtained via 'namei'.
1855      <1>      ; 3) The file is opened by 'iopen'.
1856      <1>      ; 4) Next housekeeping is performed on the fsp table
1857      <1>      ; and the user's open file list - u.fp.
1858      <1>      ; a) u.fp and fsp are scanned for the next available slot.
1859      <1>      ; b) An entry for the file is created in the fsp table.
1860      <1>      ; c) The number of this entry is put on u.fp list.
1861      <1>      ; d) The file descriptor index to u.fp list is pointed
1862      <1>      ; to by u.r0.
1863      <1>      ;
1864      <1>      ; Calling sequence:
1865      <1>      ; sysopen; name; mode
1866      <1>      ; Arguments:
1867      <1>      ; name - file name or path name
1868      <1>      ; mode - 0 to open for reading
1869      <1>      ; 1 to open for writing
1870      <1>      ; Inputs: (arguments)
1871      <1>      ; Outputs: *u.r0 - index to u.fp list (the file descriptor)
1872      <1>      ; is put into r0's location on the stack.
1873      <1>      ; .....
1874      <1>      ;
1875      <1>      ; Retro UNIX 8086 v1 modification:
1876      <1>      ; 'sysopen' system call has two arguments; so,
1877      <1>      ; * 1st argument, name is pointed to by BX register
1878      <1>      ; * 2nd argument, mode is in CX register
1879      <1>      ;
1880      <1>      ; AX register (will be restored via 'u.r0') will return
1881      <1>      ; to the user with the file descriptor/number
1882      <1>      ; (index to u.fp list).
1883      <1>      ;
1884      <1>      ; call arg2
1885      <1>      ; * name - 'u.namep' points to address of file/path name
1886      <1>      ; in the user's program segment ('u.segmt')
1887      <1>      ; with offset in BX register (as sysopen argument 1).
1888      <1>      ; * mode - sysopen argument 2 is in CX register
1889      <1>      ; which is on top of stack.
1890      <1>      ;
1891      <1>      ; jsr r0,arg2 / get sys args into u.namep and on stack
1892      <1>      ;
1893      <1>      ; ; system call registers: ebx, ecx (through 'sysenter')
1894      <1>      ;
1895      <1>      ; TRDOS 386 (05/10/2016)
1896      <1>      ;
1897      <1>      ; INPUT ->
1898      <1>      ; CL = File Access Value (Open Mode)
1899      <1>      ; 0 - Open file for reading
1900      <1>      ; 1 - Open file for writing
1901      <1>      ; 2 - Open device for reading
1902      <1>      ; 3 - Open device for writing
1903      <1>      ; EBX = Pointer to filename/devicename (ASCIIIZ)
1904      <1>      ; OUTPUT ->
1905      <1>      ; eax = File/Device Handle/Number (index) (AL)
1906      <1>      ; cf = 1 -> Error code in AL
1907      <1>      ;
1908      <1>      ; Modified Registers: EAX (at the return of system call)
1909      <1>      ;
1910      <1>      ;
1911      <1>      cmp     cl, 1 ; read file (0), write file (1)
1912      <1>      jna     short sysopen_0
1913      <1>      ;
1914      <1>      ; 17/04/2021 (temporary)
1915      <1>      cmp     cl, 3
1916      <1>      jna     sysopen_device
1917      <1>      ;
1918      <1>      ; Invalid access code
1919      <1>      mov     eax, ERR_INV_PARAMETER
1920      <1>      jmp     sysopen_dev_err
1921      <1>      ;
1922      <1>      sysopen_dev_err:
1923      <1>      mov     [u.r0], eax
1924      <1>      mov     [u.error], eax
1925      <1>      jmp     error
1926      <1>      ;
1927      <1>      sysopen_0:
1928      <1>      mov     [u.namep], ebx
1929      <1>      push    ecx
1930      <1>      mov     esi, ebx
1931      <1>      ; file name is forced, change directory as temporary
1932      <1>      mov     ax, 1
1933      <1>      mov     [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
1934      <1>      call    set_working_path
1935      <1>      call    set_working_path_x ; 17/10/2016
1936      <1>      jnc     short sysopen_1
1937      <1>      ;
1938      <1>      syscreat_err: ; ecx = file attributes (for 'syscreat')
1939      <1>      pop     ecx ; open mode
1940      <1>      and     eax, eax ; 0 -> Bad Path!
1941      <1>      jnz     short sysopen_err
1942      <1>      ; eax = 0
1943      <1>      mov     eax, ERR_DIR_NOT_FOUND ; Directory not found !
1944      <1>      sysopen_err:
1945      <1>      mov     [u.r0], eax
1946      <1>      mov     [u.error], eax
1947      <1>      call    reset_working_path
1948      <1>      jmp     error
1949      <1>      ;
1950      <1>      sysopen_1:
1951      <1>      mov     esi, FindFile_Name
1952      <1>      mov     ax, 1800h ; Only files
1953      <1>      call    find_first_file

```

```

1954 0000D018 5A      <1>    pop     edx
1955 0000D019 72E0    <1>    jc      short sysopen_err ; eax = 2 (File not found !)
1956                                <1>
1957                                <1>    ; check_open_file_attr_access_code
1958                                <1>
1959 0000D01B F6C307    <1>    test    bl, 7 ; system, hidden, readonly
1960 0000D01E 740B    <1>    jz      short sysopen_2
1961                                <1>
1962 0000D020 20D2    <1>    and     dl, dl ; 0 = read mode
1963 0000D022 7407    <1>    jz      short sysopen_2
1964                                <1>
1965                                <1> sysopen_access_err:
1966                                <1>    ; 1 = write, 2 = read & write, >2 = invalid
1967 0000D024 B80B000000    <1>    mov     eax, ERR_FILE_ACCESS ; 11 = 'permission denied !'
1968 0000D029 EBD0    <1>    jmp     short sysopen_err
1969                                <1>
1970                                <1> sysopen_2:
1971                                <1>    ; esi = Directory Entry (FindFile_DirEntry) Location
1972                                <1>    ; mov     ebx, esi
1973                                <1>    ; 03/09/2024
1974 0000D02B 89FB    <1>    mov     ebx, edi ; Directory entry in directory buffer
1975 0000D02D 31F6    <1>    xor     esi, esi ; 0
1976 0000D02F 31FF    <1>    xor     edi, edi ; 0
1977                                <1> sysopen_3: ; scan the list of entries in fsp table
1978 0000D031 80BE[5E8E0100]00 <1>    cmp     byte [esi+u.fp], 0
1979 0000D038 760E    <1>    jna     short sysopen_4 ; empty slot
1980                                <1>    ; inc     si
1981                                <1>    ; 19/08/2024
1982 0000D03A 46      <1>    inc     esi
1983 0000D03B 6683FE0A    <1>    cmp     si, 10
1984 0000D03F 72F0    <1>    jb      short sysopen_3
1985                                <1> toomanyf:
1986 0000D041 B80D000000    <1>    mov     eax, ERR_TOO_MANY_FILES ; too many open files !
1987 0000D046 EBB3    <1>    jmp     short sysopen_err
1988                                <1>
1989                                <1> sysopen_4:
1990 0000D048 80BF[04840100]00 <1>    cmp     byte [edi+OF_MODE], 0 ; scan open files table
1991 0000D04F 7609    <1>    jna     short sysopen_5
1992                                <1>
1993                                <1>    ; inc     di
1994                                <1>    ; 19/08/2024
1995 0000D051 47      <1>    inc     edi
1996 0000D052 6683FF20    <1>    cmp     di, OPENFILES ; max. number of open files (sytem)
1997 0000D056 72F0    <1>    jb      short sysopen_4
1998 0000D058 EBE7    <1>    jmp     short toomanyf
1999                                <1>
2000                                <1> sysopen_5:
2001                                <1>    ; 19/08/2024
2002                                <1>    ;;;
2003 0000D05A 50      <1>    push    eax ; size
2004 0000D05B 668B4314    <1>    mov     ax, [ebx+DirEntry_FstClusHI]
2005 0000D05F C1E010    <1>    shl     eax, 16
2006 0000D062 668B431A    <1>    mov     ax, [ebx+DirEntry_FstClusLO]
2007 0000D066 5B      <1>    pop     ebx ; size
2008                                <1>    ; eax = First Cluster
2009                                <1>    ; ebx = File Size
2010 0000D067 80FA01    <1>    cmp     dl, 1 ; is open mode = open for write ?
2011 0000D06A 7535    <1>    jne     short sysopen_8
2012                                <1>    ; check if the file is already open for write
2013 0000D06C 57      <1>    push    edi
2014 0000D06D 29FF    <1>    sub     edi, edi
2015 0000D06F 8A15[02800100] <1>    mov     dl, [FindFile_Drv]
2016                                <1> sysopen_5_@:
2017 0000D075 80BF[04840100]02 <1>    cmp     byte [edi+OF_MODE], 2 ; open for write
2018 0000D07C 7519    <1>    jne     short sysopen_5_@@@
2019 0000D07E 3A97[E4830100] <1>    cmp     dl, [edi+OF_DRIVE]
2020 0000D084 7511    <1>    jne     short sysopen_5_@@@
2021 0000D086 C1E702    <1>    shl     edi, 2
2022 0000D089 3B87[64830100] <1>    cmp     eax, [edi+OF_FCLUSTER]
2023 0000D08F 7503    <1>    jne     short sysopen_5_@@
2024 0000D091 5F      <1>    pop     edi
2025 0000D092 EB90    <1>    jmp     short sysopen_access_err
2026                                <1> sysopen_5_@@:
2027 0000D094 C1EF02    <1>    shr     edi, 2
2028                                <1>    ;;;
2029                                <1> sysopen_5_@@@:
2030 0000D097 47      <1>    inc     edi
2031 0000D098 6683FF20    <1>    cmp     di, OPENFILES
2032 0000D09C 72D7    <1>    jb      short sysopen_5_@
2033 0000D09E 5F      <1>    pop     edi
2034 0000D09F B201    <1>    mov     dl, 1 ; open for write
2035                                <1> sysopen_8:
2036                                <1>    inc     dl
2037 0000D0A3 8897[04840100] <1>    mov     [edi+OF_MODE], dl
2038 0000D0A9 8A15[02800100] <1>    mov     dl, [FindFile_Drv]
2039 0000D0AF 8897[E4830100] <1>    mov     [edi+OF_DRIVE], dl ; Logical DOS drive number
2040                                <1>    ; shl     di, 2 ; *4 (dword offset)
2041                                <1>    ; 23/07/2022
2042 0000D0B5 C1E702    <1>    shl     edi, 2
2043                                <1>
2044                                <1>    ; 19/08/2024
2045                                <1>    ; mov     [edi+OF_SIZE], eax ; File size in bytes
2046 0000D0B8 899F[E4840100] <1>    mov     [edi+OF_SIZE], ebx ; File size in bytes
2047                                <1>
2048                                <1>    ; 19/08/2024
2049                                <1>    ; eax = First Cluster
2050                                <1>    ; mov     ax, [ebx+DirEntry_FstClusHI]
2051                                <1>    ; shl     eax, 16
2052                                <1>    ; mov     ax, [ebx+DirEntry_FstClusLO]
2053                                <1>
2054 0000D0BE 8987[64830100] <1>    mov     [edi+OF_FCLUSTER], eax ; First cluster
2055 0000D0C4 8987[E4860100] <1>    mov     [edi+OF_CCLUSTER], eax ; Current cluster
2056                                <1>
2057 0000D0CA 31DB    <1>    xor     ebx, ebx
2058 0000D0CC 899F[64840100] <1>    mov     [edi+OF_POINTER], ebx ; offset pointer (0)
2059 0000D0D2 899F[64870100] <1>    mov     [edi+OF_CCINDEX], ebx ; cluster index (0)
2060                                <1>
2061 0000D0D8 A1[74800100] <1>    mov     eax, [FindFile_DirFirstCluster]
2062 0000D0DD 8987[64850100] <1>    mov     [edi+OF_DIRFCLUSTER], eax
2063                                <1>
2064 0000D0E3 A1[78800100] <1>    mov     eax, [FindFile_DirCluster]
2065 0000D0E8 8987[E4850100] <1>    mov     [edi+OF_DIRCLUSTER], eax
2066                                <1>
2067                                <1>    ; Get (& Save) volume ID
2068                                <1>    ; Important for files of removable drives
2069                                <1>    ; (In order to check the drive has same volume/disk)
2070 0000D0EE 88D7    <1>    mov     bh, dl
2071 0000D0F0 81C300010900 <1>    add     ebx, Logical_DOSDisks
2072                                <1>
2073 0000D0F6 8A4303    <1>    mov     al, [ebx+LD_FATType]
2074                                <1>
2075                                <1>    ; 19/12/2025
2076                                <1>    %if 0
2077                                <1>    cmp     al, 1

```



```

2078      <1>      jb      short sysopen_6_fs
2079      <1> %endif
2080      <1>      cmp     al, 2
2081      <1>      ja      short sysopen_6_fat32
2082      <1> sysopen_6_fat:
2083      <1>      mov     eax, [ebx+LD_BPB+VolumeID]
2084      <1>      jmp     short sysopen_7
2085      <1>
2086      <1>      ; 19/12/2025
2087      <1> %if 0
2088      <1> sysopen_6_fs:
2089      <1>      mov     eax, [ebx+LD_FS_VolumeSerial]
2090      <1>      jmp     short sysopen_7
2091      <1> %endif
2092      <1>
2093      <1> sysopen_6_fat32:
2094      <1>      mov     eax, [ebx+LD_BPB+FAT32_VolID]
2095      <1> sysopen_7:
2096      <1>      mov     [Current_VolSerial], eax
2097      <1>
2098      <1>      mov     [edi+OF_VOLUMEID], eax
2099      <1>
2100      <1>      ; 24/10/2016
2101      <1>      ; shr     di, 1 ; 4/2, word offset
2102      <1>      ; 23/07/2022
2103      <1>      shr     edi, 1
2104      <1>      mov     bx, [FindFile_DirEntryNumber]
2105      <1>      mov     [edi+OF_DIRENTRY], bx
2106      <1>
2107      <1>      xor     edx, edx
2108      <1>      ; shr     di, 2 ; /4 (byte offset)
2109      <1>      ; shr     di, 1 ; 2/2, byte offset
2110      <1>      ; 23/07/2022
2111      <1>      shr     edi, 1
2112      <1>      mov     byte [edi+OF_OPENCOUNT], dl ; 0
2113      <1>      mov     byte [edi+OF_STATUS], dl ; 0
2114      <1>
2115      <1>      mov     ebx, edi
2116      <1>      inc     bl
2117      <1>
2118      <1>      mov     [esi+u.fp], bl ; Open File Entry Number
2119      <1>      mov     [u.r0], esi ; move index to u.fp list
2120      <1>      ; into eax on stack
2121      <1>
2122      <1>      call    reset_working_path
2123      <1>
2124      <1>      jmp     sysret
2125      <1>
2126      <1>
2127      <1>      ; fsp table (original UNIX v1)
2128      <1>
2129      <1>      ; Entry
2130      <1>      ;      15                                0
2131      <1>      ;      1 |-----| i-number of open file |-----|
2132      <1>      ;      | r/w |-----| device number |-----|
2133      <1>      ;      |-----|
2134      <1>      ;      |-----|
2135      <1>      ;      (*) |-----| offset pointer, i.e., r/w pointer to file |-----|
2136      <1>      ;      |-----|
2137      <1>      ;      |-----| flag that says | number of processes |
2138      <1>      ;      |-----| file deleted | that have file open |-----|
2139      <1>      ;      2 |-----|
2140      <1>      ;      |-----|
2141      <1>      ;      |-----|
2142      <1>      ;      |-----|
2143      <1>      ;      |-----|
2144      <1>      ;      |-----|
2145      <1>      ;      |-----|
2146      <1>      ;      |-----|
2147      <1>      ;      |-----|
2148      <1>      ;      3 |-----|
2149      <1>      ;      |-----|
2150      <1>      ;      |-----|
2151      <1>      ;      |-----|
2152      <1>      ; (*) Retro UNIX 386 v1 modification: 32 bit offset pointer
2153      <1>
2154      <1>      ; 27/03/2020 - Retro UNIX 386 v2 - FSP (OPEN FILES) TABLE
2155      <1>
2156      <1>      ; Entry
2157      <1>      ;      15                                7                                0
2158      <1>      ;      1 |-----| i-number of open file |-----| |
2159      <1>      ;      |-----| high word of 32 bit i-number |-----|
2160      <1>      ;      |-----| open mode & status | device number |-----|
2161      <1>      ;      |-----| reserved byte | open count |-----|
2162      <1>      ;      |-----| offset pointer, i.e., r/w pointer to file |-----|
2163      <1>      ;      |-----| 64 bit file offset pointer (bit 16-31) |-----|
2164      <1>      ;      |-----| 64 bit file offset pointer (bit 32-47) |-----|
2165      <1>      ;      |-----| 64 bit file offset pointer (bit 48-63) |-----|
2166      <1>      ;      2 |-----|
2167      <1>      ;      |-----|
2168      <1>      ;      |-----|
2169      <1>      ;      |-----|
2170      <1>      ;      |-----|
2171      <1>      ;      |-----|
2172      <1>      ;      |-----|
2173      <1>      ;      |-----|
2174      <1>      ;      |-----|
2175      <1>      ;      |-----|
2176      <1>      ;      |-----|
2177      <1>      ;      |-----|
2178      <1>      ;      |-----|
2179      <1>      ;      |-----|
2180      <1>      ;      |-----|
2181      <1>      ;      |-----|
2182      <1>      ;      |-----|
2183      <1>      ;      |-----|
2184      <1>
2185      <1>      ; 23/07/2022 - TRDOS 386 kernel v2.0.5
2186      <1>      ; OPENFILES equ 10 (sysdefs.s)
2187      <1>      ; 06/10/2016
2188      <1>      ; Open File Parameters (trdoskx.s)
2189      <1>      ; OF_FCLUSTER: resd OPENFILES ; First clusters of open files
2190      <1>      ; OF_DRIVE: resb OPENFILES ; Logical DOS drive numbers of open files
2191      <1>      ; OF_MODE: resb OPENFILES ; Open mode (1 = read, 2 = write, 3 = r&w)
2192      <1>      ; OF_STATUS: resb OPENFILES ; (bit 0 = read, bit 1 = write)
2193      <1>      ; OF_OPENCOUNT: resb OPENFILES ; Open counts of open files
2194      <1>      ; OF_POINTER: resd OPENFILES ; File seek/read/write pointer
2195      <1>      ; OF_SIZE: resd OPENFILES ; File sizes of open files (in bytes)
2196      <1>      ; OF_DIRFCLUSTER: resd OPENFILES ; Directory First Clusters of open files
2197      <1>      ; OF_DIRCLUSTER: resd OPENFILES ; Directory (Entry) clusters of open files
2198      <1>      ; OF_VOLUMEID: resd OPENFILES ; Vol ID for removable drives of open files
2199      <1>      ; OF_CCLUSTER: resd OPENFILES ; Current clusters of open files
2200      <1>      ; OF_CCINDEX: resd OPENFILES ; Cluster index numbers of current clusters
2201      <1>      ; 24/10/2016

```

```

2202 <1> ;OF_DIRENTRY:      resw OPENFILES ; Directory entry index no. in dir cluster
2203 <1>
2204 <1> ; 17/04/2021
2205 <1> ; ('sysopen_device' procedure is disabled as temporary)
2206 <1>
2207 <1> ;sysopen_device:
2208 <1> ; ; 15/10/2016
2209 <1> ; ; 08/10/2016
2210 <1> ; ; 07/10/2016 (TRDOS 386 = TRDOS v2.0)
2211 <1> ; push ecx ; open mode
2212 <1> ; mov ebp, esp
2213 <1> ; mov ecx, 16 ; transfer length = 16 bytes
2214 <1> ; sub esp, ecx
2215 <1> ; mov edi, esp ; destination address
2216 <1> ; mov esi, ebx ; dev name in user's memory space
2217 <1> ; call transfer_from_user_buffer
2218 <1> ; jnc short sysopen_dev_0
2219 <1> ; ; eax = ERR_OUT_OF_MEMORY = 4 = ERR_MINOR_IM
2220 <1> ; pop ecx
2221 <1> ;sysopen_dev_err:
2222 <1> ; mov [u.r0], eax
2223 <1> ; mov [u.error], eax
2224 <1> ; jmp error
2225 <1> ;sysopen_dev_0:
2226 <1> ; mov esi, edi ; Device name addr (max. 16 bytes, ASCIIIZ)
2227 <1> ; ; for example: "tty, TTY, /dev/tty"
2228 <1> ; call get_device_number
2229 <1> ; mov esp, ebp
2230 <1> ; pop ecx
2231 <1> ; jnc short sysopen_dev_1
2232 <1> ; mov eax, ERR_INV_DEV_NAME ; 24 ; 'invalid device name !'
2233 <1> ; jmp short sysopen_dev_err
2234 <1> ;sysopen_dev_1:
2235 <1> ; ; eax = Device Number (AL)
2236 <1> ; ; cl = Open mode (2 = device read, 3 = device write)
2237 <1> ; xor ebx, ebx ; 0
2238 <1> ;sysopen_dev_2: ; scan the list of entries
2239 <1> ; cmp [ebx+u.fp], bl ; 0
2240 <1> ; jna short sysopen_dev_3 ; empty slot
2241 <1> ; inc bl
2242 <1> ; cmp bl, 10
2243 <1> ; jb short sysopen_dev_2
2244 <1> ;
2245 <1> ; mov eax, ERR_TOO_MANY_FILES ; too many open files !
2246 <1> ; jmp short sysopen_dev_err
2247 <1> ;sysopen_dev_3:
2248 <1> ; mov [u.r0], ebx ; File/Device index/handle/descriptor
2249 <1> ; ; eax = device number (entry offset)
2250 <1> ; mov ch, [eax+DEV_ACCESS] ; bit 0 = accessible by users
2251 <1> ; ; bit 1 = read access perm
2252 <1> ; ; bit 2 = write access perm
2253 <1> ; ; bit 3 = IOCTL permit to users
2254 <1> ; ; bit 4 = block device if set
2255 <1> ; ; bit 5 = 16 bit or 1024 byte
2256 <1> ; ; bit 6 = 32 bit or 2048 byte
2257 <1> ; ; bit 7 = installable device drv
2258 <1> ; test ch, 1 ; accessible by normal users (except root)
2259 <1> ; jnz short sysopen_dev_4 ; yes, permission has been given
2260 <1> ; cmp byte [u.uid], 0 ; root?
2261 <1> ; jna short sysopen_dev_4 ; superuser can open all devices
2262 <1> ;sysopen_dev_perm_err:
2263 <1> ; mov eax, ERR_DEV_ACCESS ; 11 = 'permission denied !'
2264 <1> ; jmp short sysopen_dev_err
2265 <1> ;sysopen_dev_4:
2266 <1> ; shr ch, 1 ; result: 1 = read, 2 = write, 3 = r & w
2267 <1> ; dec cl ; result: 1 = read, 2 = write
2268 <1> ; test cl, ch
2269 <1> ; jz short sysopen_dev_perm_err
2270 <1> ;
2271 <1> ; shl ch, 1 ; bit 0 = 0
2272 <1> ; ; eax = device number (entry offset)
2273 <1> ; call device_open
2274 <1> ; jc short sysopen_dev_perm_err
2275 <1> ;
2276 <1> ; ; eax = device number (entry offset)
2277 <1> ; or al, 80h ; set device bit (set bit 7 to 1)
2278 <1> ; mov ebx, [u.r0]
2279 <1> ; mov [ebx+u.fp], al ; bit 7 (=1) points to device
2280 <1> ;
2281 <1> ; jmp sysret
2282 <1>
2283 <1> sysmkdir: ; < make directory >
2284 <1> ; 23/07/2022 - TRDOS 386 Kernel v2.0.5
2285 <1> ; 15/10/2016
2286 <1> ; 10/10/2016 (TRDOS 386 = TRDOS v2.0)
2287 <1> ; ; -derived from INT_21H.ASM-
2288 <1> ; ; ("loc_INT21h_create_file")
2289 <1> ; ; 10/07/2011 (12/03/2011)
2290 <1> ; ; INT 21h Function AH = 3Ch
2291 <1> ; ; Create File
2292 <1> ; ; INPUT
2293 <1> ; ; CX = Attributes
2294 <1> ; ; DS:BX= Address of zero terminated path name
2295 <1> ; ;
2296 <1> ; ;
2297 <1> ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
2298 <1> ; 27/05/2013 - 02/08/2013 (Retro UNIX 8086 v1)
2299 <1> ;
2300 <1> ; 'sysmkdir' creates an empty directory whose name is
2301 <1> ; pointed to by arg 1. The mode of the directory is arg 2.
2302 <1> ; The special entries '.' and '..' are not present.
2303 <1> ; Errors are indicated if the directory already exists or
2304 <1> ; user is not the super user.
2305 <1> ;
2306 <1> ; Calling sequence:
2307 <1> ; sysmkdir; name; mode
2308 <1> ; Arguments:
2309 <1> ; name - points to the name of the directory
2310 <1> ; mode - mode of the directory
2311 <1> ; Inputs: (arguments)
2312 <1> ; Outputs: -
2313 <1> ; (sets 'directory' flag to 1;
2314 <1> ; 'set user id on execution' and 'executable' flags to 0)
2315 <1> ; .....
2316 <1> ;
2317 <1> ; Retro UNIX 8086 v1 modification:
2318 <1> ; 'sysmkdir' system call has two arguments; so,
2319 <1> ; * 1st argument, name is pointed to by BX register
2320 <1> ; * 2nd argument, mode is in CX register
2321 <1> ;
2322 <1> ; TRDOS 386 (10/10/2016)
2323 <1> ;
2324 <1> ; INPUT ->
2325 <1> ; CL = Directory Attributes

```

```

2326      <1>      ;          bit 0 (1) - Read only file/dir (R)
2327      <1>      ;          bit 1 (1) - Hidden file/dir (H)
2328      <1>      ;          ;          bit 2 (1) - System file/dir (R)
2329      <1>      ;          ;          bit 3 (1) - Volume label/name (V)
2330      <1>      ;          ;          bit 4 (1) - Subdirectory (D)
2331      <1>      ;          ;          bit 5 (1) - File/Dir has been archived (A)
2332      <1>      ;          ;          CX = 0 -> create normal directory
2333      <1>      ;          ;          EBX = Pointer to directory name (ASCIIZ) -path-
2334      <1>      ;
2335      <1>      ; OUTPUT ->
2336      <1>      ;          eax = First cluster of the new directory
2337      <1>      ;          cf = 1 -> Error code in AL
2338      <1>      ;
2339      <1>      ; Modified Registers: EAX (at the return of system call)
2340      <1>      ;
2341      <1>      ; Note: If the file or directory is existing
2342      <1>      ;          an access error will be returned.
2343      <1>      ;
2344      <1>      and    cx, cx ; if cx = 0 -> create a normal subdir
2345      <1>      jz     short sysmkdir_1
2346      <1>      ;
2347      <1>      test   cl, 10h ; if dir flags set, also use other flags
2348      <1>      ;jnz    sysmkdir_0 ; jump to head of 'syscreat'
2349      <1>      ; 23/07/2022
2350      <1>      jz     short sysmkdir_invf
2351      <1>      sysmkdir_3:
2352      <1>      jmp     sysmkdir_0
2353      <1>      ;
2354      <1>      sysmkdir_invf:
2355      <1>      ; CX has wrong flags
2356      <1>      mov     eax, ERR_INV_FLAGS
2357      <1>      jmp     sysopen_dev_err
2358      <1>      ;
2359      <1>      sysmkdir_1:
2360      <1>      mov     cl, 10h ; set subdir flag and reset other flags
2361      <1>      jmp     sysmkdir_0
2362      <1>      ; 23/07/2022
2363      <1>      jmp     short sysmkdir_3 ; jump to head of 'syscreat'
2364      <1>      sysmkdir_2:
2365      <1>      ; jump from 'syscreat' ; from 'syscreat_1'
2366      <1>      ; CL = Directory attributes/flags
2367      <1>      mov     esi, FindFile_Name
2368      <1>      call    make_sub_directory
2369      <1>      ;jc      sysopen_err ; NOTE: Old type (TRDOS 8086)
2370      <1>      ;          ; error codes must be modified
2371      <1>      ;          ; for next TRDOS 386 versions
2372      <1>      ;          ; (10/10/2016)
2373      <1>      ;          ; Old (MSDOS type)
2374      <1>      ;          ; error codes (2011):
2375      <1>      ;          ; 2 = file not found
2376      <1>      ;          ; 3 = directory not found
2377      <1>      ;          ; 5 = access denied
2378      <1>      ;          ; 12 = no more files
2379      <1>      ;          ; 19 = disk write protected
2380      <1>      ;          ; 39 = insufficient disk space
2381      <1>      ;          ; 'sysdefs.s' ; 10/10/2016
2382      <1>      ; 23/07/2022
2383      <1>      jnc     short sysmkdir_4
2384      <1>      jmp     sysopen_err
2385      <1>      ;
2386      <1>      sysmkdir_4:
2387      <1>      mov     [u.r0], eax ; New sub dir's first cluster
2388      <1>      call    reset_working_path
2389      <1>      jmp     sysret
2390      <1>      ;
2391      <1>      sysclose: ;<close file>
2392      <1>      ; 23/07/2022 - TRDOS 386 v2.0.5
2393      <1>      ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
2394      <1>      ;
2395      <1>      ; 14/05/2015 (Retro UNIX 386 v1 - Beginning)
2396      <1>      ; 22/05/2013 - 26/05/2013 (Retro UNIX 8086 v1)
2397      <1>      ;
2398      <1>      ; 'sysclose', given a file descriptor in 'u.r0', closes the
2399      <1>      ; associated file. The file descriptor (index to 'u.fp' list)
2400      <1>      ; is put in r1 and 'fclose' is called.
2401      <1>      ;
2402      <1>      ; Calling sequence:
2403      <1>      ;          sysclose
2404      <1>      ; Arguments:
2405      <1>      ;          -
2406      <1>      ; Inputs: *u.r0 - file descriptor
2407      <1>      ; Outputs: -
2408      <1>      ; .....
2409      <1>      ;
2410      <1>      ; Retro UNIX 8086 v1 modification:
2411      <1>      ; The user/application program puts file descriptor
2412      <1>      ; in BX register as 'sysclose' system call argument.
2413      <1>      ; (argument transfer method 1)
2414      <1>      ;
2415      <1>      ; TRDOS 386 (06/10/2016)
2416      <1>      ;
2417      <1>      ; INPUT ->
2418      <1>      ;          EBX = File Handle/Number (file index) (AL)
2419      <1>      ; OUTPUT ->
2420      <1>      ;          cf = 0 -> EAX = 0
2421      <1>      ;          cf = 1 -> Error code in EAX (ERR_FILE_NOT_OPEN)
2422      <1>      ;
2423      <1>      ; Modified Registers: EAX (at the return of system call)
2424      <1>      ;
2425      <1>      ;
2426      <1>      mov     eax, ebx
2427      <1>      xor     ebx, ebx
2428      <1>      mov     [u.r0], ebx ; 0 ; return value of EAX
2429      <1>      call    fclose
2430      <1>      ;jnc     sysret
2431      <1>      ; 23/07/2022
2432      <1>      jc      short sysclose_err
2433      <1>      jmp     sysret
2434      <1>      sysclose_err:
2435      <1>      ; 24/04/2025
2436      <1>      seektell_err:
2437      <1>      device_rw_err:
2438      <1>      mov     eax, ERR_FILE_NOT_OPEN ; file not open !
2439      <1>      mov     [u.error], eax ;
2440      <1>      mov     [u.r0], eax ; ! invalid handle !
2441      <1>      jmp     error
2442      <1>      ;
2443      <1>      sysread: ; < read from file >
2444      <1>      ; 27/09/2024
2445      <1>      ; 18/09/2024
2446      <1>      ; 03/09/2024 (TRDOS v2.0.9)
2447      <1>      ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2448      <1>      ; -derived from INT_21H.ASM-
2449      <1>      ; ("loc_INT21h_read_file")

```

```

2450      <1>      ; 13/03/2011 (05/03/2011)
2451      <1>      ; INT 21h Function AH = 3Fh
2452      <1>      ; Read from a File
2453      <1>      ; INPUT
2454      <1>      ; BX = File Handle
2455      <1>      ; CX = Number of bytes to read
2456      <1>      ; DS:DX= Buffer address
2457      <1>      ;
2458      <1>      ; Note: TRDOS 386 'sysread' has been derived from
2459      <1>      ; Retro UNIX 386 v1 'sysread', except a few
2460      <1>      ; code modifications.
2461      <1>      ;
2462      <1>      ; 13/05/2015 (Retro UNIX 386 v1)
2463      <1>      ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
2464      <1>      ; 23/05/2013 (Retro UNIX 8086 v1)
2465      <1>      ;
2466      <1>      ; 'sysread' is given a buffer to read into and the number of
2467      <1>      ; characters to be read. If finds the file from the file
2468      <1>      ; descriptor located in *u.r0 (r0). This file descriptor
2469      <1>      ; is returned from a successful open call (sysopen).
2470      <1>      ; The i-number of file is obtained via 'rw1' and the data
2471      <1>      ; is read into core via 'readi'.
2472      <1>      ;
2473      <1>      ; Calling sequence:
2474      <1>      ; sysread; buffer; nchars
2475      <1>      ; Arguments:
2476      <1>      ; buffer - location of contiguous bytes where
2477      <1>      ; input will be placed.
2478      <1>      ; nchars - number of bytes or characters to be read.
2479      <1>      ; Inputs: *u.r0 - file descriptor (& arguments)
2480      <1>      ; Outputs: *u.r0 - number of bytes read.
2481      <1>      ; .....
2482      <1>      ;
2483      <1>      ; Retro UNIX 8086 v1 modification:
2484      <1>      ; 'sysread' system call has three arguments; so,
2485      <1>      ; * 1st argument, file descriptor is in BX register
2486      <1>      ; * 2nd argument, buffer address/offset in CX register
2487      <1>      ; * 3rd argument, number of bytes is in DX register
2488      <1>      ;
2489      <1>      ; AX register (will be restored via 'u.r0') will return
2490      <1>      ; to the user with number of bytes read.
2491      <1>      ;
2492      <1>      ; TRDOS 386 (05/10/2016)
2493      <1>      ;
2494      <1>      ; INPUT ->
2495      <1>      ; EBX = File handle (descriptor/index)
2496      <1>      ; ECX = Buffer address
2497      <1>      ; EDX = Number of bytes
2498      <1>      ; OUTPUT ->
2499      <1>      ; EAX = Number of bytes have been read
2500      <1>      ; cf = 1 -> Error code in AL
2501      <1>      ;
2502      <1>      ; Modified Registers: EAX (at the return of system call)
2503      <1>      ;
2504      <1>      ;
2505      <1>      ; EBX = File descriptor
2506      <1>      call getf1
2507      <1>      jc short device_read ; read data from device
2508      <1>      ;
2509      <1>      ; EAX = First cluster of the file
2510      <1>      ;
2511      <1>      call rw1 ; 03/09/2024 (major modification)
2512      <1>      jnc short sysread_0
2513      <1>      ;
2514      <1>      sysrw_err: ; 03/09/2024
2515      <1>      mov [u.r0], eax ; error code
2516      <1>      jmp error
2517      <1>      ;
2518      <1>      sysread_0:
2519      <1>      call readi
2520      <1>      jmp short rw0
2521      <1>      ;
2522      <1>      syswrite: ; < write to file >
2523      <1>      ; 27/09/2024
2524      <1>      ; 18/09/2024
2525      <1>      ; 03/09/2024
2526      <1>      ; 25/08/2024 - TRDOS 386 v2.0.9
2527      <1>      ; 23/10/2016
2528      <1>      ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2529      <1>      ; -derived from INT_21H.ASM-
2530      <1>      ; ("loc_INT21h_write_file")
2531      <1>      ;
2532      <1>      ; 13/03/2011 (05/03/2011)
2533      <1>      ; INT 21h Function AH = 40h
2534      <1>      ; Write to a File
2535      <1>      ; INPUT
2536      <1>      ; BX = File Handle
2537      <1>      ; CX = Number of bytes to write
2538      <1>      ; DS:DX= Buffer address
2539      <1>      ;
2540      <1>      ; Note: TRDOS 386 'syswrite' has been derived from
2541      <1>      ; Retro UNIX 386 v1 'syswrite', except a few
2542      <1>      ; code modifications.
2543      <1>      ;
2544      <1>      ; 13/05/2015 (Retro UNIX 386 v1)
2545      <1>      ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
2546      <1>      ; 23/05/2013 (Retro UNIX 8086 v1)
2547      <1>      ;
2548      <1>      ; 'syswrite' is given a buffer to write onto an output file
2549      <1>      ; and the number of characters to write. If finds the file
2550      <1>      ; from the file descriptor located in *u.r0 (r0). This file
2551      <1>      ; descriptor is returned from a successful open or create call
2552      <1>      ; (sysopen or syscreat). The i-number of file is obtained via
2553      <1>      ; 'rw1' and buffer is written on the output file via 'write'.
2554      <1>      ;
2555      <1>      ; Calling sequence:
2556      <1>      ; syswrite; buffer; nchars
2557      <1>      ; Arguments:
2558      <1>      ; buffer - location of contiguous bytes to be writtten.
2559      <1>      ; nchars - number of characters to be written.
2560      <1>      ; Inputs: *u.r0 - file descriptor (& arguments)
2561      <1>      ; Outputs: *u.r0 - number of bytes written.
2562      <1>      ; .....
2563      <1>      ;
2564      <1>      ; Retro UNIX 8086 v1 modification:
2565      <1>      ; 'syswrite' system call has three arguments; so,
2566      <1>      ; * 1st argument, file descriptor is in BX register
2567      <1>      ; * 2nd argument, buffer address/offset in CX register
2568      <1>      ; * 3rd argument, number of bytes is in DX register
2569      <1>      ;
2570      <1>      ; AX register (will be restored via 'u.r0') will return
2571      <1>      ; to the user with number of bytes written.
2572      <1>      ;
2573      <1>      ; INPUT ->

```

```

2574      ;      EBX = File handle (descriptor/index)
2575      ;      ECX = Buffer address
2576      ;      EDX = Number of bytes
2577      ; OUTPUT ->
2578      ;      EAX = Number of bytes have been written
2579      ;      cf = 1 -> Error code in AL
2580      ;
2581      ; Modified Registers: EAX (at the return of system call)
2582      ;
2583      ;
2584      ; EBX = File descriptor
2585 0000D1D0 E8BB300000      call    getf1
2586 0000D1D5 721B          jc     short device_write ; write data to device
2587      ; EAX = First cluster of the file
2588      ; EBX = File number (Open file number) ; 23/10/2016
2589      ;
2590 0000D1D7 E818000000      call    rw1      ; 03/09/2024 (major modification)
2591      ; 03/09/2024
2592 0000D1DC 72E1          jc     short sysrw_err
2593      ;
2594      ; 25/08/2024 (*)
2595      ; jnc short syswrite_0
2596      ; mov [u.r0], eax ; error code
2597      ; jmp error
2598      ;
2599      ; 25/08/2024 - TRDOS 386 v2.0.9 (bugfix) (*)
2600      ; (if eax = 0, lets add a cluster at 'mget_w_0')
2601      ; ('mget_w' and 'add_new_cluster' procs are modified)
2602      ;
2603      syswrite_0:
2604 0000D1DE E86C3B0000      call    writei ; 24/08/2024 ('mget_w' modification)
2605      ;
2606 0000D1E3 A1[808E0100]    rw0: ; 1:      mov    eax, [u.nread]
2607 0000D1E8 A3[548E0100]    mov    [u.r0], eax
2608 0000D1ED E98CF8FFFF      jmp     sysret
2609      ;
2610      ; 17/04/2021 (temporary)
2611      device_write:
2612      device_read:
2613      ; 26/09/2024
2614      ; 18/09/2024 - TRDOS 386 v2.0.9
2615      ; 17/04/2021 - TRDOS 386 v2.0.4
2616      ; (temporary modifications)
2617      ;
2618      ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2619      ; cl = DEV_OPENMODE ; open mode
2620      ; ch = DEV_ACCESS ; access flags
2621      ; al = DEV_DRIVER ; device number (eax)
2622      ;
2623      ; 24/04/2025 - TRDOS 386 v2.0.10
2624      ; 18/09/2024 (temporary)
2625      ; call rw2 ; file not open ; cf = 1
2626      ; jmp error
2627      ; 26/09/2024
2628      ; jmp short sysrw_err
2629      ; 24/04/2025
2630 0000D1F2 EBA9          jmp     short device_rw_err
2631      ;
2632      ;
2633      ; test cl, 1 ; 1 = read, 2 = write, 3 = read&write
2634      ; jz short rw3
2635      ;
2636      ; mov ebx, eax
2637      ; shl bx, 2 ; *4
2638      ;
2639      ; test ch, 80h ; bit 7, installable device driver flag
2640      ; jz short d_read_2 ; kernel device
2641      ; ; installable device
2642      ; d_read_1:
2643      ; jmp dword [ebx+IDEV_RADDR-4]
2644      ; d_read_2:
2645      ; jmp dword [ebx+KDEV_RADDR-4]
2646      ;
2647      ; device_write:
2648      ; 17/04/2021 - TRDOS 386 v2.0.4
2649      ; (temporary modifications)
2650      ;
2651      ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2652      ; cl = DEV_OPENMODE ; open mode
2653      ; ch = DEV_ACCESS ; access flags
2654      ; al = DEV_DRIVER ; device number (eax)
2655      ;
2656      ; 17/04/2021 (temporary)
2657      ; jmp short rw2 ; file not open
2658      ;
2659      ; test cl, 2 ; 1 = read, 2 = write, 3 = read&write
2660      ; jz short rw3
2661      ;
2662      ; mov ebx, eax
2663      ; shl bx, 2 ; *4
2664      ;
2665      ; test ch, 80h ; bit 7, installable device driver flag
2666      ; jz short d_write_2 ; kernel device
2667      ; ; installable device
2668      ; d_write_1:
2669      ; jmp dword [ebx+IDEV_WADDR-4]
2670      ; d_write_2:
2671      ; jmp dword [ebx+KDEV_WADDR-4]
2672      ;
2673      ; rw1:
2674      ; 27/09/2024
2675      ; 03/09/2024 (TRDOS 386 v2.0.9)
2676      ; 17/04/2021 (TRDOS 386 v2.0.4)
2677      ; 11/10/2016 (TRDOS 386 = TRDOS v2.0)
2678      ; 14/05/2015 (Retro UNIX 386 v1)
2679      ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
2680      ; 23/05/2013 - 24/05/2013 (Retro UNIX 8086 v1)
2681      ; System call registers: ebx, ecx, edx (through 'sysenter')
2682      ;
2683      ; EBX = File descriptor
2684      ; call getf1 ; calling point in 'getf' from 'rw1'
2685      ; jc short device_rw ; read/write data from/to device
2686      ; EAX = First cluster of the file
2687      ;
2688      ; 03/09/2024
2689 0000D1F4 09C0          or     eax, eax
2690 0000D1F6 7421          jz     short rw7 ; eax = 0 -> empty file (OK for now)
2691      ;
2692 0000D1F8 83F802          cmp    eax, 2 ; is it valid cluster number ?
2693      ; jb short rw2
2694      ; 03/09/2024
2695 0000D1FB 7307          jnb    short rw6 ; yes, check upper limit
2696      ;
2697      ;;;

```

```

2698      <1>      ; eax = 1 -> invalid cluster number
2699      <1>      rw5:
2700      0000D1FD B823000000      <1>      mov     eax, ERR_CLUSTER ; 35 ; 'cluster not available !'
2701      0000D202 EB2C           <1>      jmp     short rw4 ; cf = 1
2702      <1>
2703      <1>      rw6:; 03/09/2024 (check cluster number is valid or not)
2704      0000D204 53             <1>      push    ebx
2705      <1>      ; ebx <= OPENFILES-1 ; 0-31
2706      0000D205 8ABB[E4830100] <1>      mov     bh, [ebx+OF_DRIVE] ; drive number * 256
2707      0000D20B 28DB           <1>      sub     b1, b1 ; 27/09/2024
2708      0000D20D 8B9B78010900 <1>      mov     ebx, [ebx+Logical_DOSDisks+LD_Clusters]
2709      0000D213 43             <1>      inc     ebx ; cluster count + 1 = last cluster number
2710      0000D214 39C3           <1>      cmp     ebx, eax
2711      0000D216 5B             <1>      pop     ebx
2712      0000D217 72E4           <1>      jb     short rw5
2713      <1>      rw7:
2714      <1>      ;;
2715      <1>
2716      0000D219 890D[788E0100] <1>      mov     [u.base], ecx ; buffer address/offset
2717      <1>      ;(in the user's virtual memory space)
2718      0000D21F 8915[7C8E0100] <1>      mov     [u.count], edx
2719      0000D225 C705[C08E0100]0000- <1>      mov     dword [u.error], 0 ; reset the last error code
2720      0000D22F C3             <1>      retn
2721      <1>
2722      <1>      ; 24/04/2025 - TRDOS 386 v2.0.10
2723      <1>      %if 0
2724      <1>
2725      <1>      rw2:
2726      <1>      mov     eax, ERR_FILE_NOT_OPEN ; file not open !
2727      <1>      ;mov     dword [u.error], eax
2728      <1>      ;retn
2729      <1>
2730      <1>      %endif
2731      <1>      ; 03/09/2024
2732      <1>      ; 17/04/2021
2733      <1>      ;jmp     short rw4
2734      <1>
2735      <1>      ; 03/09/2024
2736      <1>      rw3:
2737      <1>      ; mov     eax, ERR_FILE_ACCESS ; permission denied !
2738      <1>      ; stc
2739      <1>
2740      <1>      rw4:; 17/04/2021
2741      0000D230 A3[C08E0100] <1>      mov     dword [u.error], eax
2742      0000D235 C3             <1>      retn
2743      <1>
2744      <1>      systimer:
2745      <1>      ; 23/07/2022 - TRDOS 386 kernel v2.0.5
2746      <1>      ; 02/01/2017
2747      <1>      ; 21/12/2016
2748      <1>      ; 19/12/2016
2749      <1>      ; 10/12/2016 (callback)
2750      <1>      ; 10/06/2016
2751      <1>      ; 07/06/2016
2752      <1>      ; 06/06/2016
2753      <1>      ; 21/05/2016
2754      <1>      ; 19/05/2016
2755      <1>      ; 18/05/2016 - TRDOS 386 (TRDOS v2.0)
2756      <1>      ; (TRDOS 386 feature only!)
2757      <1>
2758      <1>      ; (start or stop timer event(s))
2759      <1>
2760      <1>      INPUT ->
2761      <1>      ; BL = signal return byte (response byte)
2762      <1>      ; (Any requested value between 0 and 255)
2763      <1>      ; (kernel will put it at the requested address)
2764      <1>      ; BH = time count unit
2765      <1>      ; 0 = stop timer event
2766      <1>      ; 1 = 18.2 ticks per second
2767      <1>      ; 2 = 10 milliseconds
2768      <1>      ; 3 = 1 second (for real time clock interrupt)
2769      <1>      ; 4 = time/tick count in current time count unit
2770      <1>      ; // 10/12/2016
2771      <1>      ; 80h = stop timer event (callback method)
2772      <1>      ; 81h = 18.2 ticks per second, callback method
2773      <1>      ; 82h = 10 milliseconds, callback method
2774      <1>      ; 83h = 1 second (for RTC int), callback method
2775      <1>      ; 84h = current time count unit, callback method
2776      <1>
2777      <1>      ; Note: Only 03h or 83h will set real time clock
2778      <1>      ; (RTC) events (Others are for PIT events)!
2779      <1>
2780      <1>      ; NOTE: If callback (user service) method is used,
2781      <1>      ; EDX will point to the return address (of service
2782      <1>      ; procedure) in user's space instead of signal
2783      <1>      ; response byte address. (TRDOS 386 kernel will
2784      <1>      ; direct the cpu to that address -in user's space-
2785      <1>      ; at the return of system call or interrupt
2786      <1>      ; just after the adjusted count/time is elapsed.)
2787      <1>      ; User's service routine must be ended with a
2788      <1>      ; 'iret'. Normal return addresses from system
2789      <1>      ; calls or and interrupts will be kept same except
2790      <1>      ; the timer returns.
2791      <1>
2792      <1>      ;
2793      <1>      ; BH = 0 -> stop timer event
2794      <1>      ; BL = timer event number (1 to 255) if BH = 0
2795      <1>      ; If BL = 0, all timer events (which are belongs
2796      <1>      ; to running process) will be stopped
2797      <1>      ; ECX = Time/Tick count (depending on time count unit)
2798      <1>      ; EDX = signal return (Response) byte address
2799      <1>      ; (virtual address in user's memory space)
2800      <1>      ; OUTPUT ->
2801      <1>      ; AL = timer event number (1 to 255) (max. value = 16)
2802      <1>      ; IF BH Input = 0 & CF = 0 & AL = 0 ->
2803      <1>      ; timer event(s) has/have been stopped/finished
2804      <1>      ; CF = 1 & AL = 0 -> no timer setting space to set
2805      <1>      ; CF = 1 & AL > 0 -> timer count unit is not usable
2806      <1>
2807      <1>      ; NOTE: To modify a time count for a user function,
2808      <1>      ; at first, current timer event must be stopped
2809      <1>      ; then a new timer event (which is related with
2810      <1>      ; same user function) must be started.
2811      <1>
2812      <1>      ; signal return (response) byte may be used for
2813      <1>      ; several purposes. kernel will put this value
2814      <1>      ; to requested address during timer interrupt,
2815      <1>      ; program/user can check this value to understand
2816      <1>      ; which event has been occurred and what is changed.
2817      <1>      ; (Multi timer events can share same signal address)
2818      <1>
2819      <1>      ; NOTE: If the process is running while the time count
2820      <1>      ; is reached, kernel will put signal return (response)
      <1>      ; byte value at requested address during timer

```

```

2821      <1>      ;      interrupt and the process will continue to run.
2822      <1>      ;      Program/process must call (jump to) it's timer event
2823      <1>      ;      function as required, for checking the timer event
2824      <1>      ;      status via signal return (response) byte address.
2825      <1>      ;
2826      <1>      ;      If the process is not running (waiting or sleeping
2827      <1>      ;      or released) while the time count is reached,
2828      <1>      ;      it is restarted from where it left, to ensure
2829      <1>      ;      proper multi media (video, audio, clock, timer)
2830      <1>      ;      functionality.
2831      <1>      ;
2832      <1>      ;      (It is better to use 'syswait' or 'sysleep',
2833      <1>      ;      or 'sysrele' system call just after the timer
2834      <1>      ;      function. Otherwise, timer events may block other
2835      <1>      ;      processes which are not using timer events.)
2836      <1>      ;
2837      <1>      ; Timer Event Structure: (max. 16 timer events, 16*16 bytes)
2838      <1>      ; Owner:      resb 1 ; 0 = free
2839      <1>      ;      ;>0 = process number (u.uno)
2840      <1>      ; Callback:   resb 1 ; 1 = callback, 0 = response byte
2841      <1>      ; Interrupt:   resb 1 ; 0 = Timer interrupt (or none)
2842      <1>      ;      ; 1 = Real Time Clock interrupt
2843      <1>      ; Response:    resb 1 ; 0 to 255, signal return value
2844      <1>      ; Count Limit: resd 1 ; count of ticks (total/set)
2845      <1>      ; Current Count: resd 1 ; count of ticks (current)
2846      <1>      ; Response Addr: resd 1 ; response byte (pointer) address
2847      <1>      ;
2848      <1>      ;
2849      <1>      ; 19/12/2016 (timer callback)
2850      0000D236 C605[3C880100]00 <1>      mov     byte [tcallback], 0
2851      0000D23D C605[3D880100]00 <1>      mov     byte [trtc], 0
2852      0000D244 C705[C88E0100]0000- <1>      mov     dword [u.tcb], 0 ; this is not necessary...
2853      0000D24C 0000 <1>      ;
2854      0000D24E 80FF80 <1>      cmp     bh, 80h
2855      0000D251 7224 <1>      jnb     short systimer_cb2
2856      0000D253 7704 <1>      ja      short systimer_cb0
2857      <1>      ;
2858      0000D255 31D2 <1>      xor     edx, edx ; 0, reset callback address
2859      0000D257 EB0B <1>      jmp     short systimer_cb1
2860      <1>      ;
2861      <1>      systimer_cb0:
2862      0000D259 80FF84 <1>      cmp     bh, 84h
2863      0000D25C 7764 <1>      ja      short systimer_5 ; undefined, error
2864      <1>      ;
2865      <1>      ;mov     byte [tcallback], 1 ; 19/12/2016
2866      0000D25E FE05[3C880100] <1>      inc     byte [tcallback]
2867      <1>      ;
2868      <1>      systimer_cb1:
2869      0000D264 0FB635[A58E0100] <1>      movzx   esi, byte [u.uno] ; process number
2870      <1>      ;shl     si, 2
2871      <1>      ; 23/07/2022
2872      0000D26B C1E602 <1>      shl     esi, 2
2873      0000D26E 8996[E48D0100] <1>      mov     [esi+p.tcb-4], edx ; set process timer callback address
2874      <1>      ; (overwrite prev value if it is set!)
2875      0000D274 80E77F <1>      and     bh, 7Fh
2876      <1>      ;
2877      <1>      systimer_cb2:
2878      0000D277 80FF02 <1>      cmp     bh, 2
2879      0000D27A 7446 <1>      je      short systimer_5 ; only 18.2 ticks per second is usable
2880      <1>      ; 10 milliseconds (100 Hertz) timer
2881      <1>      ; will be set later (18/05/2016)
2882      0000D27C 774C <1>      ja      short systimer_6
2883      <1>      ;
2884      0000D27E 20FF <1>      and     bh, bh
2885      <1>      ;jz     systimer_9 ; stop timer event(s)
2886      <1>      ; 23/07/2022
2887      0000D280 7505 <1>      jnz     short systimer_19
2888      0000D282 E9BA000000 <1>      jmp     systimer_9
2889      <1>      ;
2890      <1>      ; bh = 1 (timer interrupt, 18.2 Hz, IBM PC/AT ROMBIOS default)
2891      <1>      ;
2892      <1>      systimer_19:
2893      0000D287 B00A <1>      mov     al, 10 ; (*)
2894      <1>      ;
2895      <1>      systimer_0:
2896      0000D289 B710 <1>      mov     bh, 16
2897      <1>      ;
2898      0000D28B 383D[33830100] <1>      cmp     [timer_events], bh ; 16 ; 07/06/2016
2899      0000D291 7319 <1>      jnb     short systimer_3 ; max. 16 timer events
2900      <1>      ;
2901      0000D293 50 <1>      push    eax ; (*)
2902      <1>      ;
2903      0000D294 BF[688F0100] <1>      mov     edi, timer_set ; beginning address of timer events
2904      <1>      ; setting space
2905      0000D299 30C0 <1>      xor     al, al ; 0
2906      <1>      systimer_1:
2907      0000D29B FEC0 <1>      inc     al
2908      0000D29D 803F00 <1>      cmp     byte [edi], 0 ; is it free space ?
2909      0000D2A0 7639 <1>      jna     short systimer_7 ; yes
2910      0000D2A2 FECF <1>      dec     bh
2911      0000D2A4 7405 <1>      jz      short systimer_2
2912      0000D2A6 83C710 <1>      add     edi, 16
2913      0000D2A9 EBF0 <1>      jmp     short systimer_1 ; next event space
2914      <1>      ;
2915      <1>      systimer_2:
2916      0000D2AB 58 <1>      pop     eax ; (*) discard
2917      <1>      systimer_3:
2918      0000D2AC C605[548E0100]00 <1>      mov     byte [u.r0], 0
2919      <1>      systimer_4:
2920      0000D2B3 C705[C08E0100]1B00- <1>      mov     dword [u.error], ERR_MISC
2921      0000D2BB 0000 <1>      ; one of miscellaneous/other errors
2922      0000D2BD E99CF7FFFF <1>      jmp     error ; cf -> 1
2923      <1>      ;
2924      <1>      systimer_5:
2925      0000D2C2 883D[548E0100] <1>      mov     [u.r0], bh ; Time count unit (=2 or >3)
2926      0000D2C8 EBE9 <1>      jmp     short systimer_4 ; 07/06/2016
2927      <1>      ;
2928      <1>      systimer_6:
2929      0000D2CA 80FF04 <1>      cmp     bh, 4
2930      0000D2CD 77F3 <1>      ja      short systimer_5 ; undefined time count unit
2931      <1>      ;jnb     short systimer_16
2932      <1>      ;
2933      <1>      ;mov     al, 1 ; default (use current timer unit)
2934      <1>      ; countdown value is in ECX !
2935      <1>      ; max. value of ecx = 4294967296/10
2936      <1>      ;jmp     short systimer_0
2937      <1>      ;jmp     short systimer_19
2938      0000D2CF 74B6 <1>      je      short systimer_19
2939      <1>      ;
2940      <1>      systimer_16:
2941      <1>      ; bh = 3
2942      <1>      ; timer event via real time clock interrupt

```

```

2943      <1>      ; interrupt/update frequency: 1 Hz (1 tick per second)
2944      <1>
2945      0000D2D1 B0B6      <1>      mov     al, 182 ; (*) ; 18.2 * 10
2946      0000D2D3 FE05[3D880100] <1>      inc     byte [trtc] ; timer event via real time clock
2947      0000D2D9 EBAE      <1>      jmp      short systimer_0
2948      <1>
2949      <1> systimer_7:
2950      0000D2DB A2[548E0100] <1>      mov     [u.r0], al ; timer event number
2951      <1>      ;
2952      <1>      ; edi = address of empty timer event area
2953      0000D2E0 A0[A58E0100] <1>      mov     al, [u.uno]
2954      0000D2E5 FA        <1>      cli      ; disable interrupts
2955      0000D2E6 AA        <1>      stosb    ; process number
2956      0000D2E7 A0[3C880100] <1>      mov     al, [tcallback] ; timer callback flag
2957      0000D2EC AA        <1>      stosb    ; 1= callback method, 0= signal response byte method
2958      0000D2ED A0[3D880100] <1>      mov     al, [trtc] ; timer interrupt type
2959      0000D2F2 AA        <1>      stosb    ; 1= real time clock, 0= programmable interval timer
2960      0000D2F3 88D8      <1>      mov     al, bl ; Signal return (Response) value
2961      0000D2F5 AA        <1>      stosb    ; response byte
2962      0000D2F6 58        <1>      pop     eax ; (*) ; 10 or 182
2963      0000D2F7 89D3      <1>      mov     ebx, edx ; virtual address for response/signal byte
2964      0000D2F9 F7E1      <1>      mul     ecx
2965      <1>      ; (eax = 10 * count of 18.2 Hz timer ticks)
2966      <1>      ; (count down step = 10)
2967      0000D2FB AB        <1>      stosd    ; count limit (reset value)
2968      0000D2FC AB        <1>      stosd    ; current count value
2969      <1>
2970      <1>      ; 19/12/2016
2971      0000D2FD 803D[3C880100]00 <1>      cmp     byte [tcallback], 0 ; timer callback method ?
2972      0000D304 7604      <1>      jna      short systimer_17 ; no
2973      0000D306 89D8      <1>      mov     eax, ebx ; virtual address for callback routine
2974      0000D308 EB0D      <1>      jmp      short systimer_18
2975      <1>
2976      <1> systimer_17: ; signal response byte method
2977      <1>      ; ebx = virtual address
2978      <1>      ; [u.pgdir] = page directory's physical address
2979      <1>      ; 20/02/2017
2980      0000D30A FE05[3E880100] <1>      inc     byte [no_page_swap] ; 1
2981      <1>      ; Do not add this page to swap queue
2982      <1>      ; and remove it from swap queue if it is
2983      <1>      ; on the queue.
2984      0000D310 E8EF88FFFF <1>      call    get_physical_addr
2985      0000D315 721A      <1>      jc       short systimer_8 ; 07/06/2016
2986      <1>      ; eax = physical address of the virtual address in user's space
2987      <1> systimer_18:
2988      0000D317 AB        <1>      stosd    ; response addr (physical) or callback addr (virtual)
2989      0000D318 FE05[33830100] <1>      inc     byte [timer_events] ; 07/06/201
2990      <1>      ; 02/01/2017
2991      0000D31E 0FB605[A58E0100] <1>      movzx   eax, byte [u.uno]
2992      0000D325 FE80[D78D0100] <1>      inc     byte [eax+p.timer-1]
2993      <1>      ;
2994      0000D32B FB        <1>      sti      ; enable interrupts
2995      0000D32C E94DF7FFFF <1>      jmp      sysret
2996      <1>
2997      <1> systimer_8:
2998      <1>      ; 10/06/2016
2999      <1>      ; 07/06/2016
3000      0000D331 28C0      <1>      sub     al, al ; 0
3001      0000D333 8847F4      <1>      mov     [edi-12], al ; clear process number (free timer event)
3002      <1>      ;mov    dword [edi], eax ; 0
3003      0000D336 FB        <1>      sti      ;
3004      0000D337 A2[548E0100] <1>      mov     [u.r0], al ; 0
3005      0000D33C E91DF7FFFF <1>      jmp      error
3006      <1>
3007      <1> systimer_9:
3008      <1>      ; 10/06/2016
3009      <1>      ; 07/06/2016
3010      0000D341 28C0      <1>      sub     al, al
3011      0000D343 A2[548E0100] <1>      mov     byte [u.r0], al ; 0
3012      0000D348 3805[33830100] <1>      cmp     byte [timer_events], al ; 0
3013      0000D34E 7631      <1>      jna      short systimer_12
3014      <1>
3015      <1>      ; Note: ecx and edx are undefined here
3016      <1>      ; (for stop timer function)
3017      <1>
3018      0000D350 BE[688F0100] <1>      mov     esi, timer_set ; beginning address of timer events
3019      <1>      ; setting space
3020      0000D355 A0[A58E0100] <1>      mov     al, [u.uno]
3021      <1>
3022      0000D35A B710      <1>      mov     bh, 16
3023      <1>
3024      0000D35C 08DB      <1>      or      bl, bl
3025      0000D35E 7544      <1>      jnz     short systimer_15
3026      <1>
3027      <1>      ; clear timer event areas belong to current process
3028      <1>      ; (for stopping all timer events belong to current process)
3029      0000D360 FA        <1>      cli      ; disable interrupts
3030      <1> systimer_10:
3031      <1>      ; 10/06/2016
3032      <1>      ; 07/06/2016
3033      0000D361 8A26      <1>      mov     ah, [esi]
3034      0000D363 08E4      <1>      or      ah, ah ; 0 ?
3035      0000D365 7411      <1>      jz       short systimer_11
3036      0000D367 38C4      <1>      cmp     ah, al ; is the process number (owner) same ?
3037      0000D369 750D      <1>      jne      short systimer_11 ; no
3038      <1>
3039      <1>      ;mov    byte [esi], 0
3040      0000D36B 66C7060000 <1>      mov     word [esi], 0 ; clear
3041      <1>      ;mov    dword [esi+12], 0 ; clear
3042      <1>
3043      0000D370 FE0D[33830100] <1>      dec     byte [timer_events]
3044      0000D376 7409      <1>      jz       short systimer_12
3045      <1>
3046      <1> systimer_11:
3047      0000D378 FECF      <1>      dec     bh
3048      0000D37A 7405      <1>      jz       short systimer_12
3049      0000D37C 83C610      <1>      add     esi, 16
3050      0000D37F EBE0      <1>      jmp      short systimer_10
3051      <1>
3052      <1> systimer_12:
3053      0000D381 0FB635[A58E0100] <1>      movzx   esi, byte [u.uno]
3054      0000D388 08DB      <1>      or      bl, bl ; all timer events or one timer event ?
3055      0000D38A 740C      <1>      jz       short systimer_13
3056      0000D38C 8A9E[D78D0100] <1>      mov     bl, [esi+p.timer-1]
3057      0000D392 20DB      <1>      and     bl, bl ; previous number of timer events for the process
3058      0000D394 7408      <1>      jz       short systimer_14
3059      0000D396 FECB      <1>      dec     bl ; previous number of timer events for the process - 1
3060      <1> systimer_13:
3061      0000D398 889E[D78D0100] <1>      mov     [esi+p.timer-1], bl ; 0 ; no timer events for process
3062      <1> systimer_14:
3063      0000D39E FB        <1>      sti      ; enable interrupts
3064      0000D39F E9DAF6FFFF <1>      jmp      sysret
3065      <1>
3066      <1> systimer_15:

```



```

3067 0000D3A4 38FB      <1>      cmp     b1, bh ; 16
3068                  <1>      ;ja systimer_4      ; max. 16 timer events !
3069                  <1>      ; 23/07/2022
3070 0000D3A6 7605      <1>      jna     short systimer_21
3071                  <1>      systimer_20:
3072 0000D3A8 E906FFFFFF <1>      jmp     systimer_4
3073                  <1>      systimer_21:      ; 23/07/2022
3074 0000D3AD 88DA      <1>      mov     dl, b1
3075 0000D3AF FECA      <1>      dec     dl ; 16 -> 15 ... 1 -> 0
3076 0000D3B1 C0E204    <1>      shl     dl, 4 ; * 16
3077 0000D3B4 0FB6FA    <1>      movzx   edi, dl
3078 0000D3B7 01F7      <1>      add     edi, esi ; timer_set
3079                  <1>
3080 0000D3B9 3A07      <1>      cmp     al, [edi] ; process number
3081                  <1>      ;jne     systimer_4
3082                  <1>      ; 23/07/2022
3083 0000D3BB 75EB      <1>      jne     short systimer_20 ; jmp systimer_4
3084                  <1>
3085                  <1>      ; same process ID
3086 0000D3BD FA        <1>      cli     ; disable interrupts
3087                  <1>      ; 10/06/2016 ; 02/01/2017
3088                  <1>      ;mov     byte [edi], 0
3089 0000D3BE 66C7070000 <1>      mov     word [edi], 0 ; clear
3090                  <1>      ;mov     dword [edi+12], 0 ; clear
3091 0000D3C3 FE0D[33830100] <1>      dec     byte [timer_events]
3092 0000D3C9 EBB6      <1>      jmp     short systimer_12
3093                  <1>
3094                  <1>      sysvideo: ; VIDEO DATA TRANSFER FUNCTIONS
3095                  <1>      ; 11/08/2022
3096                  <1>      ; 08/08/2022
3097                  <1>      ; 23/07/2022 (TRDOS 386 v2.0.5)
3098                  <1>      ; 06/03/2021
3099                  <1>      ; 02/03/2021
3100                  <1>      ; 28/02/2021
3101                  <1>      ; 27/02/2021
3102                  <1>      ; 26/02/2021
3103                  <1>      ; 25/02/2021
3104                  <1>      ; 21/02/2021, 22/02/2021, 23/02/2021
3105                  <1>      ; 15/02/2021, 16/02/2021, 18/02/2021
3106                  <1>      ; 10/02/2021, 11/02/2021, 12/02/2021
3107                  <1>      ; 07/02/2021, 08/02/2021
3108                  <1>      ; 01/02/2021, 02/02/2021, 05/02/2021
3109                  <1>      ; 29/01/2021, 30/01/2021, 31/01/2021
3110                  <1>      ; 23/01/2021, 24/01/2021, 28/01/2021
3111                  <1>      ; 18/01/2021, 19/01/2021, 22/01/2021
3112                  <1>      ; 04/01/2021, 10/01/2021, 11/01/2021
3113                  <1>      ; 01/01/2021, 02/01/2021, 03/01/2021
3114                  <1>      ; 28/12/2020, 29/12/2020, 30/12/2020
3115                  <1>      ; 25/12/2020, 26/12/2020
3116                  <1>      ; 21/12/2020, 23/12/2020
3117                  <1>      ; 12/12/2020, 14/12/2020
3118                  <1>      ; 10/12/2020, 11/12/2020
3119                  <1>      ; 03/12/2020, 04/12/2020
3120                  <1>      ; 22/11/2020, 23/11/2020
3121                  <1>      ; 21/11/2020 (TRDOS 386 v2.0.3)
3122                  <1>      ; 12/05/2017
3123                  <1>      ; 11/07/2016
3124                  <1>      ; 13/06/2016
3125                  <1>      ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
3126                  <1>
3127                  <1>      VIDEO DATA TRANSFER FUNCTIONS:
3128                  <1>
3129                  <1>      Inputs:
3130                  <1>      ; 07/02/2021
3131                  <1>      BH = 0 = VIDEO BIOS Mode 3, tty/text mode data transfers
3132                  <1>      BL =
3133                  <1>      Bits 0&1, Transfer direction
3134                  <1>      0 - System to system
3135                  <1>      1 - User to system
3136                  <1>      2 - System to user
3137                  <1>      3 - Exchange (Swap) - 28/01/2021
3138                  <1>      Bits 2, Transfer Type
3139                  <1>      0 - Display page (complete) transfer
3140                  <1>      1 - Display page window (col,row) transfer
3141                  <1>      ; 28/01/2021
3142                  <1>      Bits 3..7 - Reserved, undefined (must be 0)
3143                  <1>      ; 28/01/2021
3144                  <1>      /// BL = 0 -> System to system (display page) transfer
3145                  <1>      CL = Source page (0FFh = current video page)
3146                  <1>      DL = Destination page (0FFh = current video page)
3147                  <1>      (Note: Nothing to do if src & dest are same page)
3148                  <1>      /// BL = 1&2 -> user to system & system to user transfer
3149                  <1>      ECX = User's buffer address
3150                  <1>      DL = Video page (0FFh = current video page)
3151                  <1>      /// BL = 3 -> exchange (swap) display page ; 28/01/2021
3152                  <1>      ECX = User's buffer address
3153                  <1>      DL = Video page (0FFh = current video page)
3154                  <1>      EDI = Swap address in user's memory (must be > 0)
3155                  <1>      /// BL = 5&6&7 -> user to system, system to user transfer
3156                  <1>      (system window is in current/active display page)
3157                  <1>      ESI = User's buffer address
3158                  <1>      ECX Low 16 bits = Top left column (X1 position)
3159                  <1>      ECX High 16 bits = Top row (Y1 position)
3160                  <1>      EDX Low 16 bits = Bottom right column (X2 position)
3161                  <1>      EDX High 16 bits = Bottom row (Y2 position)
3162                  <1>      ;
3163                  <1>      If BL = 5 or BL bit 0 & bit 1 are 1 ; 28/01/2021
3164                  <1>      EDI = Swap address (in user's memory space)
3165                  <1>      (If swap address > 0, previous content of the window
3166                  <1>      will be saved into swap area in user's memory space)
3167                  <1>      /// BL = 4 -> system to system transfer
3168                  <1>      ESI = System's source buffer (video page) address
3169                  <1>      ECX Low 16 bits = Top left column (X1 position)
3170                  <1>      ECX High 16 bits = Top row (Y1 position)
3171                  <1>      EDX Low 16 bits = Bottom right column (X2 position)
3172                  <1>      EDX High 16 bits = Bottom row (Y2 position)
3173                  <1>      EDI = System's destination buffer (video page) addr
3174                  <1>
3175                  <1>      ; 06/02/2021
3176                  <1>      ; 05/02/2021
3177                  <1>      ; 01/02/2021, 02/02/2021
3178                  <1>      ; 30/01/2021, 31/02/2021
3179                  <1>      ; 29/01/2021 (major modification)
3180                  <1>      ; 23/11/2020 (major modification)
3181                  <1>      ; 22/11/2020 (bugfixes and extensions)
3182                  <1>      BH = 1 = VGA Graphics (0A0000h) data transfers
3183                  <1>      BL bit 7
3184                  <1>      resolution (screen width) option
3185                  <1>      0 = 320 pixels
3186                  <1>      1 = 640 pixels
3187                  <1>      .. followings are same with SVGA transfer function
3188                  <1>      BL bit 6
3189                  <1>      direction option
3190                  <1>      0 = user to system (video memory)
3190                  <1>      1 = system to user

```

```

3191      <1>      ;
3192      <1>      ;
3193      <1>      ;
3194      <1>      ;
3195      <1>      ;
3196      <1>      ;
3197      <1>      ;
3198      <1>      ;
3199      <1>      ;
3200      <1>      ;
3201      <1>      ;
3202      <1>      ;
3203      <1>      ;
3204      <1>      ;
3205      <1>      ;
3206      <1>      ;
3207      <1>      ;
3208      <1>      ;
3209      <1>      ;
3210      <1>      ;
3211      <1>      ;
3212      <1>      ;
3213      <1>      ;
3214      <1>      ;
3215      <1>      ;
3216      <1>      ;
3217      <1>      ;
3218      <1>      ;
3219      <1>      ;
3220      <1>      ;
3221      <1>      ;
3222      <1>      ;
3223      <1>      ;
3224      <1>      ;
3225      <1>      ;
3226      <1>      ;
3227      <1>      ;
3228      <1>      ;
3229      <1>      ;
3230      <1>      ;
3231      <1>      ;
3232      <1>      ;
3233      <1>      ;
3234      <1>      ;
3235      <1>      ;
3236      <1>      ;
3237      <1>      ;
3238      <1>      ;
3239      <1>      ;
3240      <1>      ;
3241      <1>      ;
3242      <1>      ;
3243      <1>      ;
3244      <1>      ;
3245      <1>      ;
3246      <1>      ;
3247      <1>      ;
3248      <1>      ;
3249      <1>      ;
3250      <1>      ;
3251      <1>      ;
3252      <1>      ;
3253      <1>      ;
3254      <1>      ;
3255      <1>      ;
3256      <1>      ;
3257      <1>      ;
3258      <1>      ;
3259      <1>      ;
3260      <1>      ;
3261      <1>      ;
3262      <1>      ;
3263      <1>      ;
3264      <1>      ;
3265      <1>      ;
3266      <1>      ;
3267      <1>      ;
3268      <1>      ;
3269      <1>      ;
3270      <1>      ;
3271      <1>      ;
3272      <1>      ;
3273      <1>      ;
3274      <1>      ;
3275      <1>      ;
3276      <1>      ;
3277      <1>      ;
3278      <1>      ;
3279      <1>      ;
3280      <1>      ;
3281      <1>      ;
3282      <1>      ;
3283      <1>      ;
3284      <1>      ;
3285      <1>      ;
3286      <1>      ;
3287      <1>      ;
3288      <1>      ;
3289      <1>      ;
3290      <1>      ;
3291      <1>      ;
3292      <1>      ;
3293      <1>      ;
3294      <1>      ;
3295      <1>      ;
3296      <1>      ;
3297      <1>      ;
3298      <1>      ;
3299      <1>      ;
3300      <1>      ;
3301      <1>      ;
3302      <1>      ;
3303      <1>      ;
3304      <1>      ;
3305      <1>      ;
3306      <1>      ;
3307      <1>      ;
3308      <1>      ;
3309      <1>      ;
3310      <1>      ;
3311      <1>      ;
3312      <1>      ;
3313      <1>      ;
3314      <1>      ;

BL bit 5
masked/direct (non-masked) operations
1 = masked, 0 = non-masked (direct)
BL bit 4
page/window option
1 = window, 0 = display page (screen)
BL bit 0 to 3 (pixel operation types)
0 = copy pixels (colors) ((mask color))
1 = Change (New, Fill) color
2 = Add color
3 = Sub color
4 = OR color
5 = AND color
6 = XOR color
7 = NOT color
8 = NEG color
9 = INC color
10 = DEC color
11 = Mix (Average) colors
12 = Replace pixel colors
13 = Copy pixel block(s)
14 = Write line(s)
15 = Write character (font)

Input Registers for pixel operations:
Same with LFB data transfer function below

; 25/02/2021
; 05/02/2021, 06/02/2021
; 01/02/2021, 02/02/2021
; 30/01/2021, 31/02/2021
; 29/01/2021 (major modification)
; 23/11/2020 (major modification)
; 22/11/2020 (bugfixes and extensions)
BH = 2 = Super VGA, LINEAR FRAME BUFFER data transfers
BL bit 7
unused (invalid), must be 0
BL bit 6
direction option
0 = user to system (video memory)
1 = system to user
BL bit 5
masked/direct (non-masked) operations
1 = masked, 0 = non-masked (direct)
BL bit 4
page/window option
1 = window, 0 = display page (screen)
BL bit 0 to 3 (pixel operation types)
0 = copy pixels (colors) ((mask color))
1 = Change (New, Fill) color
2 = Add color
3 = Sub color
4 = OR color
5 = AND color
6 = XOR color
7 = NOT color
8 = NEG color
9 = INC color
10 = DEC color
11 = Mix (Average) colors
12 = Replace pixel colors
13 = Copy pixel block(s)
14 = Write line(s)
15 = Write character (font)

Note: If HW of EBX > 0, it is VESA VBE mode number
otherwise, function will be applied
to current (VESA VBE) video mode.

Input Registers for pixel operations:
-- user to system & system to system --
-- (BL = 0 to 0Fh) -- non-masked, screen --
-- (BL = 10h to 1Fh) -- non-masked, window --
-- (BL = 20h to 2Fh) -- masked, screen --
-- (BL = 30h to 3Fh) -- masked, window --
(*) window, (**) masked (***) sys to sys
for BL bit 0 to 3
00h: COPY PIXELS
If BL bit 4 = 0 ; 21/02/2021
full screen copy
ECX & EDX will not be used
(user buffer must fit to display page)
If BL bit 4 = 1 ; 21/02/2021
ECX = start position (row, column) (*)
(HW = row, CX = column)
EDX = size (rows, columns) (*)
(HW = rows, DX = columns)
(0 -> invalid)
(1 -> horizontal or vertical line)
ESI = user's buffer address
EDI = mask color (**); 25/02/2021
(this color will be excluded)
01h: CHANGE PIXEL COLORS
02h: ADD PIXEL COLORS
03h: SUB PIXEL COLORS
04h: OR PIXEL COLORS
05h: AND PIXEL COLORS
06h: XOR PIXEL COLORS
0Bh: MIX PIXEL COLORS
CL = color (8 bit, 256 colors)
ECX = color (16 bit and true colors)
EDX = start position (row, column) (*)
(HW = row, DX = column)
ESI = size (rows, columns) (*)
(HW = rows, SI = columns)
EDI = mask color (**); 25/02/2021
(this color will be excluded)
07h: NOT PIXEL COLORS
08h: NEG PIXEL COLORS
09h: INC PIXEL COLORS
0Ah: DEC PIXEL COLORS
ECX = start position (row, column) (*)
(HW = row, CX = column)
EDX = size (rows, columns) (*)
(HW = rows, DX = columns)
(0 -> invalid)
(1 -> horizontal or vertical line)
EDI = mask color (**); 25/02/2021
(this color will be excluded)
0Ch: REPLACE PIXEL COLORS
CL = current color (8 bit, 256 colors)
ECX = current color (16 bit and true colors)
DL = new color (8 bit, 256 colors)
EDX = new color (16 bit and true colors)

```

```

3315     <1> ; ESI = start position (row, column) (*)
3316     <1> ; (HW = row, SI = column)
3317     <1> ; EDI = size (rows, columns) (*)
3318     <1> ; (HW = rows, DI = columns)
3319     <1> ; 0Dh: COPY PIXEL BLOCK(S) -full screen-
3320     <1> ; -If BL bit 5 is 0-
3321     <1> ; ECX = start position (row, column) (*)
3322     <1> ; (HW = row, CX = column)
3323     <1> ; EDX = size (rows, columns) (*)
3324     <1> ; (HW = rows, DX = columns)
3325     <1> ; (0 -> invalid)
3326     <1> ; (1 -> horizontal or vertical line)
3327     <1> ; ESI = destination (row, column) (***)
3328     <1> ; -If BL bit 5 is 1-
3329     <1> ; CL = color (8 bit, 256 colors)
3330     <1> ; ECX = color (16 bit and true colors)
3331     <1> ; EDX = count of blocks (not bytes)
3332     <1> ; (limit: 2048 blocks)
3333     <1> ; ESI = user's buffer address
3334     <1> ; contains 64 bits block data
3335     <1> ; BLOCK ADDRESS - (row, col), dword
3336     <1> ; (first 32 bits)
3337     <1> ; BLOCK SIZE - (rows, cols), dword
3338     <1> ; (second 32 bits)
3339     <1> ; 10/02/2021
3340     <1> ; 0Eh: WRITE LINE(s) -full screen-
3341     <1> ; -If BL bit 5 is 0-
3342     <1> ; CL = color (8 bit, 256 colors)
3343     <1> ; ECX = color (16 bit and true colors)
3344     <1> ; DX = low 12 bits - size (length)
3345     <1> ; high 4 bits - direction or type
3346     <1> ; 0 - Horizontal line
3347     <1> ; 1 - Vertical line
3348     <1> ; > 1 - undefined, invalid
3349     <1> ; ESI = start position (row, column)
3350     <1> ; (HW = row, SI = column)
3351     <1> ; -If BL bit 5 is 1-
3352     <1> ; CL = color (8 bit, 256 colors)
3353     <1> ; ECX = color (16 bit and true colors)
3354     <1> ; DX = number of lines (in user buffer)
3355     <1> ; (limit: 2048 lines)
3356     <1> ; ESI = user's buffer
3357     <1> ; contains 64 bit data for lines
3358     <1> ; START POINT: 32 bit (row, col)
3359     <1> ; LENGTH: 32 bit
3360     <1> ; high 16 bits - 0
3361     <1> ; bit 0-11 - length
3362     <1> ; bit 12-15 - type (length)
3363     <1> ; 0Fh: WRITE CHARACTER (FONT)
3364     <1> ; CL = char's color (8 bit, 256 colors)
3365     <1> ; ECX = char's color (16 bit and true colors)
3366     <1> ; DL = Character's ASCII code
3367     <1> ; DH bit 0 -> font height
3368     <1> ; 0 -> 8x16 character font
3369     <1> ; 1 -> 8x8 character font
3370     <1> ; DH bit 1 & 2 -> scale
3371     <1> ; 0 = 1/1 (8 pixels per char row)
3372     <1> ; 1 = 2/1 (16 pixels per char row)
3373     <1> ; 2 = 3/1 (24 pixels per char row)
3374     <1> ; 3 = 4/1 (32 pixels per char row)
3375     <1> ; DH bit 6 -> [ufont] option (1 = use [ufont])
3376     <1> ; If DH bit 7 = 1
3377     <1> ; USER FONT (from user buffer)
3378     <1> ; DL = 0 -> 8x8 (width: 1 byte per row)
3379     <1> ; DL = 1 -> 8x16
3380     <1> ; DL = 2 -> 16x16 (width: 2 bytes)
3381     <1> ; DL = 3 -> 16x32
3382     <1> ; DL = 4 -> 24x24 (width: 3 bytes)
3383     <1> ; DL = 5 -> 24x48
3384     <1> ; DL = 6 -> 32x32 (width: 4 bytes)
3385     <1> ; DL = 7 -> 32x64
3386     <1> ; DL > 7 -> invalid (unused)
3387     <1> ; EDI = user's font buffer address
3388     <1> ; (NOTE: byte order is as row0,row1,row2..)
3389     <1> ; ESI = start position (row, column) (*)
3390     <1> ; (HW = row, SI = column)
3391     <1> ;
3392     <1> ; -- system to user --
3393     <1> ; BL (bit 0 to 7)
3394     <1> ; 40h: COPY PIXELS (full screen, display page)
3395     <1> ; EDI = user's buffer address
3396     <1> ; 41h: COPY PIXELS (window)
3397     <1> ; ECX = start position (row, column) (*)
3398     <1> ; (HW = row, CX = column)
3399     <1> ; EDX = size (rows, columns) (*)
3400     <1> ; (HW = rows, DX = columns)
3401     <1> ; (<=1 -> horizontal or vertical line)
3402     <1> ; EDI = user's buffer address
3403     <1> ;
3404     <1> ; Example: (29/01/2021)
3405     <1> ; ecx = 00400064h (start at row 64, column 100)
3406     <1> ; edx = 00320048h (size: 50 rows, 72 columns)
3407     <1> ; (end at row 114, column 172)
3408     <1> ; If video memory starts at 0A0000h
3409     <1> ; and if resolution is 320x200 (256 colors) ..
3410     <1> ; window start offset: (64*320)+100 = 20580
3411     <1> ; window size: 16072 bytes (pixels)
3412     <1> ; window end offset: 20580+16072 = 36652
3413     <1> ; window start address: 0A0000h+564h = 0A5064h
3414     <1> ;
3415     <1> ; Outputs:
3416     <1> ; EAX = transfer/byte count
3417     <1> ;
3418     <1> ; NOTE: If the source or destination address passes out of
3419     <1> ; video pages (display memory limits), data will not be transferred
3420     <1> ; and EAX will return as 0.
3421     <1> ;
3422     <1> ; 08/02/2021
3423     <1> ; 07/02/2021
3424     <1> ; 04/01/2021
3425     <1> ; PIXEL READ/WRITE (in current/active video mode)
3426     <1> ;
3427     <1> ; BH = 3 = Read/write pixel(s) -for all graphics modes-
3428     <1> ; BL =
3429     <1> ; 0 = Read pixel
3430     <1> ; 1 = Write pixel
3431     <1> ; 2 = swap pixel colors
3432     <1> ; 3 = mix pixel colors
3433     <1> ; 29/01/2021
3434     <1> ; 4 = read pixels from user defined positions
3435     <1> ; 5 = write single color pixels to user defined positions
3436     <1> ; 6 = write multi color pixels to user defined positions
3437     <1> ;
3438     <1> ; > 6 = invalid/unimplemented

```

```

3439 <1> ; .. for BL = 0 to 5
3440 <1> ; CL = color for writing pixel(s) or
3441 <1> ; ECX = color for writing pixel(s) in true color modes
3442 <1> ;
3443 <1> ; EDX = Offset from start of video memory (0A0000h)
3444 <1> ; or start of linear frame buffer
3445 <1> ;
3446 <1> ; 07/02/2021
3447 <1> ; .. for BL = 4
3448 <1> ; EDI = user's destination buffer address for pixel colors
3449 <1> ; 29/01/2021
3450 <1> ; .. for BL = 4 & 5
3451 <1> ; ESI = user's source buffer address for BL = 4 & 5
3452 <1> ; (buffer contains dword offset positions for pixels)
3453 <1> ; EDX = number of pixels
3454 <1> ; .. for BL = 6
3455 <1> ; ESI = user's buffer address for BL = 6
3456 <1> ; (buffer contains dword offset position and dword color
3457 <1> ; value for each pixel)
3458 <1> ; EDX = number of pixels
3459 <1> ;
3460 <1> ; Note:
3461 <1> ; Pixel read/write will be performed in current video mode.
3462 <1> ; If [CRT_MODE] < 0FFh, 0A0000h will be used
3463 <1> ; as video memory and limit will be 65536
3464 <1> ; (new/mix pixel color will be in CL)
3465 <1> ; if [CRT_MODE] = 0FFh (VESA VBE video mode)
3466 <1> ; LFB base address will be used as video memory
3467 <1> ; and limit will be video page size
3468 <1> ; (new/mix pixel color will be in CL)
3469 <1> ;
3470 <1> ; Outputs:
3471 <1> ; EAX = pixel color (according to BL and ECX input)
3472 <1> ; EAX = 0 (pixel color is 0 or there is an error)
3473 <1> ; (BL will return as 0FFh if there is an error)
3474 <1> ; ; 29/01/2021
3475 <1> ; EAX = number of pixels (for BL input = 4&5&6)
3476 <1> ;
3477 <1> ; DIRECT (STANDARD VGA/CGA) DISPLAY MEMORY ACCESS FUNCTIONS:
3478 <1> ;
3479 <1> ; BH = 4 = CGA direct video memory (0B8000h, 32K) access
3480 <1> ; Page directory & page tables of the user's
3481 <1> ; program will be updated to direct access to
3482 <1> ; 0B8000h (32K) video (CGA, color) memory; if
3483 <1> ; there is not a permission conflict or lock!
3484 <1> ; (User's program/process will have permission to
3485 <1> ; access locked display memory if the owner is
3486 <1> ; it's parent.)
3487 <1> ; ;
3488 <1> ; Screen width = 320
3489 <1> ;
3490 <1> ; BH = 5 = VGA direct video memory (0A0000h, 64K) access
3491 <1> ; Page directory & page tables of the user's
3492 <1> ; program will be updated to direct access to
3493 <1> ; 0A0000h (64K) video (VGA) memory; if there is not
3494 <1> ; a permission conflict or lock!
3495 <1> ; (User's program/process will have permission to
3496 <1> ; access locked display memory if the owner is
3497 <1> ; it's parent.)
3498 <1> ;
3499 <1> ; ; 23/11/2020
3500 <1> ; Screen width options = 320, 640, 800
3501 <1> ;
3502 <1> ; Outputs:
3503 <1> ; EAX = Display memory address for direct access
3504 <1> ; 0A0000h for VGA, 0B8000h for CGA
3505 <1> ; (Display memory size: 32K for CGA, 64K for VGA)
3506 <1> ; EAX = 0 if display page access permission has been denied.
3507 <1> ; (Locked!)
3508 <1> ;
3509 <1> ; LINEAR FRAME BUFFER ACCESS FUNCTIONS:
3510 <1> ;
3511 <1> ; BH = 6 = Linear Frame Buffer direct video memory access
3512 <1> ;
3513 <1> ; Page directory & page tables of the user's
3514 <1> ; program will be updated to direct access to
3515 <1> ; the configured LFB (Linear Frame Buffer) address,
3516 <1> ; if there is not a permission conflict or lock!
3517 <1> ; (User's program/process will have permission to
3518 <1> ; access locked display memory if the owner is
3519 <1> ; it's parent.)
3520 <1> ;
3521 <1> ; ; 10/12/2020
3522 <1> ; BL = 0FFh -> Direct LFB access for current video mode
3523 <1> ; BL = XXh < 0FFh -> Direct LFB access
3524 <1> ; for VESA video mode 1XXh
3525 <1> ;
3526 <1> ; Return: EAX = Linear Frame Buffer address
3527 <1> ; (EAX = 0 -> error)
3528 <1> ; If EAX > 0
3529 <1> ; EDX = Frame Buffer Size in bytes
3530 <1> ; BH = Requested Video Mode - 100h
3531 <1> ; (VESA VBE video modes)
3532 <1> ; BL = bits per pixel
3533 <1> ; 8 = 256 colors, 8
3534 <1> ; 16 = 65536 colors, 5-6(G)-5
3535 <1> ; 24 = RGB, 16M colors, 8-8-8
3536 <1> ; 32 = RGB + alpha bytes, 8-8-8-8
3537 <1> ; If BH = 0FFh
3538 <1> ; BL = VGA/CGA video mode (also EAX = 0)
3539 <1> ;
3540 <1> ; ** Function will return with EAX = 0 if the mode
3541 <1> ; is not a valid VESA VBE video mode as 1??h **
3542 <1> ;
3543 <1> ; ECX = Pixel resolution
3544 <1> ; CX = width (320, 640, 800, 1024, 1366, 1920)
3545 <1> ; High 16 bits of ECX = Height
3546 <1> ;
3547 <1> ; 23/11/2020
3548 <1> ; *** GET VIDEO MODE & LINEAR FRAME BUFFER INFO
3549 <1> ; (This function -7- also is used for VGA and CGA modes)
3550 <1> ;
3551 <1> ; BH = 7 = Get Linear Frame Buffer info
3552 <1> ;
3553 <1> ; ; 22/01/2021
3554 <1> ; ; 10/12/2020
3555 <1> ; BL = any -not used- (22/01/2021)
3556 <1> ;
3557 <1> ; Return:
3558 <1> ; EAX = Frame Buffer Address (0 = is not in use)
3559 <1> ; EDX = Frame Buffer Size in bytes
3560 <1> ; BH = Current Video Mode - 100h ; 22/01/2021
3561 <1> ; (VESA VBE video modes)
3562 <1> ; BL = bits per pixel

```

```

3563      8 = 256 colors, 8
3564      16 = 65536 colors, 5-6(G)-5
3565      24 = RGB, 16M colors, 8-8-8
3566      32 = RGB + alpha bytes, 8-8-8-8
3567      If BH = 0FFh
3568          BL = VGA/CGA video mode (also EAX = 0)
3569
3570      Note:
3571      Alpha byte will be used as virtual color index.
3572      (32 bit pixel colors.. byte 0,1,2 rgb and 3 alpha)
3573
3574      ** Function will return with EAX = 0 if the mode
3575      is not a valid VESA VBE video mode as 1??h **
3576
3577      ECX = Pixel resolution
3578      CX = width (320, 640, 800, 1024, 1366, 1920)
3579      High 16 bits of ECX = Height
3580
3581      NOTE: Each process will have it's own frame buffer
3582      address and resolution parameters in 'u' area.
3583      Then, if the current frame buffer & resolution
3584      is different, frame buffer r/w functions
3585      will use scale factor to convert process's
3586      pixel coordinates to actual screen coordinates.
3587      resolution -> dimensional scale
3588      color size -> color scale
3589      * RGB (TRUE) colors to 256 colors conversion:
3590      TRUE Colors -> 8,8,8 (R,G,B; byte 0 is R)
3591      256 colors -> 2,2,2,2 (R,G,B,L; bit 0&1 is R)
3592      bit 6&7 -> luminosity base level (0,1,2,3)
3593      bit 4&5 -> blue level (0,1,2,3)
3594      bit 2&3 -> green level (0,1,2,3)
3595      bit 0&1 -> red level (0,1,2,3)
3596      Example: total red level : luminosity + red level
3597      Luminosity base level: 0 -> 16
3598      1 -> 32
3599      2 -> 64
3600      3 -> 128
3601
3602      Color level:
3603      0 -> 0
3604      1 -> luminosity level
3605      2 -> luminosity level + 64
3606      3 -> 255
3607      Luminosity base level = min (R,G,B)
3608      if it is <16, it will be set to 16
3609      Color levels: Color values are fixed to (nearest)
3610      one of all possible set level (step) values
3611      (according to luminosity base level); then
3612      color levels are set to R-L, G-L, B-L.
3613      For example: If luminosity base level is 32
3614      all possible set values are 0, 32, 96, 255.
3615
3616      * RGB (TRUE) colors to 16 colors conversion:
3617      16 colors: R,B,G,L bits (4 bits)
3618      If any one of R,G,B >= 128 L = 1
3619      If max. value of (R,G,B) >= 32, it is 1
3620      else all color bits (R&G&B&L) are 0
3621      If the second value >= max. value / 2
3622      it is 1
3623      If the third value >= max. value / 2
3624      it is 1
3625      Example: R = 132, G = 64, B = 78
3626      L = 1, R = 1
3627      G < 66 --> G = 0
3628      B >= 66 --> B = 1
3629
3630      10/12/2020
3631      SET VIDEO MODE (& RETURN LFB INFO for VESA VBE VIDEO MODES)
3632
3633      BH = 8 = Set Video Mode
3634
3635      BL = Requested Video Mode (method)
3636      If BL = 0FFh
3637          CX = VESA VBE Video Mode
3638          ; 11/12/2020
3639          If EDX > 0 -> LFB INFO (user) buffer addr
3640          If BL < 0FFh, it is VGA/CGA video mode and
3641          CX & EDX will not be used
3642
3643      NOTE: The last VESA VBE video mode is 11Bh but
3644      TRDOS 386 will permit to set video mode upto 11Fh.
3645      Above 11Fh, from 140h to 1FEh, it will be accepted
3646      as Bochs/Plex86 emulator video mode and it will be
3647      used only if [vbe3] = 2 and detected
3648      video bios is BOCHS/PLEX86/QEMU/VIRTUALBOX vbios.
3649
3650      Outputs:
3651      EAX = Requested (Proper) video mode number + 1
3652      ("dec eax" by user will give requested video mode),
3653
3654      If BL input is 0FFh
3655      EDX = LFBINFO table/structure (in user's buffer addr)
3656      EDX = 0 -> Invalid LFBINFO (do not use it)
3657
3658      EAX = 0 -> Error (but EDX will not be changed)
3659
3660      03/12/2020
3661      VESA VBE3 VIDEO BIOS (32 bit) PROTECTED MODE INTERFACE SETTINGS
3662
3663      BH = 9 = set/get VBE3 Protected Mode Interface parameters
3664
3665      BL = 0 - Disable protected mode interface
3666      ([pmi32] = 0)
3667      Return: AL = 1
3668      BL = 1 - Enable protected mode Interface
3669      ([pmi32] = 1)
3670      Return: AL = 2
3671      BL = 2 - Get protected mode interface status
3672      Return: AL = [pmi32] + 1 (AL = 1 or 2)
3673
3674      If [vbe3] <> 3 --> AL = 0
3675
3676      ; 17/01/2021
3677      BL = 3 - Disable/Cancel restore permission to user
3678      Return: AL = 1 (if disabled) or 0
3679      BL = 4 - Enable/Give restore permission to user
3680      Return: AL = 2 (if enabled) or 0
3681      BL = 5 - Get video state save/restore status
3682      (permission status)
3683      Return: AL = Status (enabled = 1)
3684      ; 22/01/2021
3685      AH = state options ([srvso])
3686      BL = 6 - Return VESA VBE number/status
3687      Return: AX = status

```

```

3687         if AH = 2, AL > 0 : Emulator
3688             AH = 3, VESA VBE3 video bios
3689         ; 28/02/2021
3690         BL = 7 - Set true color mode as 32bpp (default)
3691             Return: AX = 32 (if VBE3)
3692         NOTE: Initial/default value is 32bpp for vbe3.
3693             Return: AX = 0 -> error
3694         BL = 8 - Set true color mode as 24bpp (default)
3695             Return: AX = 24
3696             ;Return: AX = 0 -> error
3697         BL = 9 - Return default/current true color mode
3698             Return: AX = 32 (32 bpp)
3699             Return: AX = 24 (24 bpp)
3700             Return: AX = 0 -> error (not VESA bios)
3701
3702         BL > 9 : not implemented (28/02/2021)
3703
3704         ; 19/01/2021 ([u.uid] check)
3705         Note: Enabling/Disabling are done by root ([u.uid] = 0)
3706             while [multi_tasking] = 0.
3707
3708     23/12/2020
3709     VIDEO MEMORY MAPPING:
3710         BH = 10 = Map video memory to user's buffer
3711
3712         BL = 0 : CGA memory (0B8000h) map (32K)
3713         BL = 1 : VGA memory (0A0000h) map (64K)
3714         BL = 2 : SVGA memory (LFB) map to user's buffer
3715
3716         ECX = User's buffer addr (low 12 bits will be cleared)
3717         EDX = Buffer size in bytes (if BL = 2)
3718             (will be trimmed if LFB size < EDX)
3719     Return:
3720         ; EAX = physical address of video memory (buffer)
3721         EBX = mapped (actual) size of video memory (bytes)
3722         ECX = virtual start address of user's video buffer
3723         EDX is same with EDX input
3724
3725         (Note: Memory page boundaries will be applied
3726             to buffer size and buff start addr by rounding down.
3727             Rounded size & address values must not be zero.)
3728         -Normally, it is expected to request mapping by using
3729             correct buffer size of current or desired video mode-
3730
3731         EAX = 0 -> error ! memory can not mapped to user
3732
3733     04/01/2021
3734     SET/READ COLOR PALETTE (set/read DAC color registers)
3735         ((256 colors (8bpp) VGA/CGA video hardware feature))
3736
3737         BH = 11 = Set/Read DAC color registers
3738
3739         (BL<4 Original method for std VGA video hardware)
3740         BL = 0 : Read all DAC color registers (256 colors)
3741             (6 bit colors, in RGB order)
3742         BL = 1 : Set all DAC color registers (256 colors)
3743             (6 bit colors, in RGB order)
3744         BL = 2 : Read single DAC color register
3745             (6 bit color, in RGB order)
3746             ((EAX will return with color value))
3747             CL = DAC color register (index)
3748         BL = 3 : Set/write single DAC color register
3749             (6 bit color, in RGB order, bit 6&7 are 0)
3750             ECX byte 0 - DAC color register
3751             ECX byte 1 - Red (6 bit)
3752             ECX byte 2 - Green (6 bit)
3753             ECX byte 3 - Blue (6 bit)
3754         (BL>3 Alternative method for BMP files etc.)
3755         BL = 4 : Read all DAC color registers (256 colors)
3756             (8 bit colors, in BGR order, bit 0&1 is 0)
3757         BL = 5 : Set all DAC color registers (256 colors)
3758             (8 bit colors, in BGR order, bit 0&1 is 0)
3759         BL = 6 : Read single DAC color register
3760             (8 bit color, in BGR order, bit 0&1 is 0)
3761             ((EAX will return with color value))
3762             CL = DAC color register (index)
3763         BL = 7 : Set/write single DAC color register
3764             (8 bit color, bit 0&1 are 0)
3765             ECX byte 0 - DAC color register
3766             ECX byte 1 - Blue (8 bit)
3767             ECX byte 2 - Green (8 bit)
3768             ECX byte 3 - Red (8 bit)
3769
3770         BL > 7 : invalid (not implemented)
3771
3772         if BL bit 2 is 1, 6 bit colors converted to 8 bit colors
3773             (low two bits of color bytes will be 0)
3774             ((color byte 0011111b will be converted to 11111100b))
3775             and RGB byte order will be
3776                 byte 0 - Blue (low 2 bits are 0)
3777                 byte 1 - Green (low 2 bits are 0)
3778                 byte 2 - Red (low 2 bits are 0)
3779                 byte 3 - pad (or zero byte)
3780             and 256 colors buffer size must be 256*4 = 1024 bytes
3781         if BL bit 2 is 0, 6 bit colors will be used directly
3782             (high two bits of 8 bit color bytes will be 0)
3783             ((dac color 111111b will be converted to 00111111b))
3784                 byte 0 - Red (high 2 bits are 0)
3785                 byte 1 - Green (high 2 bits are 0)
3786                 byte 2 - Blue (high 2 bits are 0)
3787             and 256 colors buffer size must be 256*3 = 768 bytes
3788
3789         ECX = User's buffer addr (256*3 = 768 bytes) or
3790             color
3791
3792     Return:
3793         ; EAX = buffer size (for BL input = 0,1,4,5)
3794             or color value (for BL input = 2,3,6,7)
3795
3796     10/01/2021
3797     SET/READ FONT DATA
3798
3799         BH = 12 = Set/Read Character Font Data
3800
3801         BL = 0 : Disable system font overwrite
3802         BL = 1 : Enable system font overwrite
3803         BL = 2 : Read system font 8x8
3804         BL = 3 : Read system font 8x14
3805         BL = 4 : Read system font 8x16
3806         BL = 5 : Read user defined font 8x8
3807         BL = 6 : Read user defined font 8x16
3808         BL = 7 : Write system font 8x8
3809         BL = 8 : Write system font 8x14
3810         BL = 9 : Write system font 8x16

```

```

3811     <1> ;          BL = 10 : Write user defined font 8x8
3812     <1> ;          BL = 11 : Write user defined font 8x16
3813     <1> ;
3814     <1> ;          BL > 11 : invalid (not implemented)
3815     <1> ;
3816     <1> ;          For BL = 1 to 11
3817     <1> ;              ECX = number of characters (<= 256)
3818     <1> ;              EDX = first character (ascii code in DL)
3819     <1> ;              ESI = user's buffer address
3820     <1> ;
3821     <1> ; Return:
3822     <1> ;     ; EAX = number of characters (ecx input)
3823     <1> ;     ; EAX = 0 -> error
3824     <1> ;     ; (EAX = 256 for BL = 0 and 1 if successful)
3825     <1> ;
3826     <1> ; Note: system font overwrite permission will be
3827     <1> ;       given if [multi_tasking] = 0
3828     <1> ;       and [u.uid] = 0 (BL = 1) ; 19/01/2021
3829     <1> ;       and if [ufont] bit 7 is 1 (BL = 7,8,9)
3830     <1> ;
3831     <1> ; 18/01/2021
3832     <1> ; SAVE/RESTORE STANDARD VGA VIDEO STATE
3833     <1> ;
3834     <1> ;     BH = 13 = Save/Restore std VGA video state
3835     <1> ;
3836     <1> ;     BL = 0 : Save VGA state (without DAC regs)
3837     <1> ;     Return: EAX = VideoStateID (>0)
3838     <1> ;     EAX = 0 (failed!)
3839     <1> ;     (size: 110 bytes for TRDOS 386 v2.0.3)
3840     <1> ;     BL = 1 : Restore VGA state (without DAC regs)
3841     <1> ;     ECX = VideoStateID (to be verified)
3842     <1> ;     Return: EAX = Restore size (>0)
3843     <1> ;     BL = 2 : Save VGA state (complete)
3844     <1> ;     Return: EAX = VideoStateID (>0)
3845     <1> ;     EAX = 0 (failed!)
3846     <1> ;     (size: 882 bytes for TRDOS 386 v2.0.3)
3847     <1> ;     BL = 3 : Restore VGA state (complete)
3848     <1> ;     ECX = VideoStateID (to be verified)
3849     <1> ;     Return: EAX = Restore size (>0)
3850     <1> ;
3851     <1> ; * Above options are for saving
3852     <1> ; *   video state to system memory
3853     <1> ; *   (location: VBE3VIDEOSTATE, 2048 bytes)
3854     <1> ;
3855     <1> ;     BL = 4 : Save VGA state (without DAC regs)
3856     <1> ;     ECX = buffer address
3857     <1> ;     Return: EAX = transfer count
3858     <1> ;     (size: 110 bytes for TRDOS 386 v2.0.3)
3859     <1> ;     ECX = buffer address
3860     <1> ;     BL = 5 : Restore VGA state (without DAC regs)
3861     <1> ;     ECX = buffer address
3862     <1> ;     Return: EAX = transfer count
3863     <1> ;     BL = 6 : Save VGA state (complete)
3864     <1> ;     ECX = buffer address
3865     <1> ;     Return: EAX = transfer count
3866     <1> ;     (size: 882 bytes for TRDOS 386 v2.0.3)
3867     <1> ;     BL = 7 : Restore VGA state (complete)
3868     <1> ;     ECX = buffer address
3869     <1> ;     Return: EAX = transfer count
3870     <1> ;
3871     <1> ; * Above options are for saving
3872     <1> ; *   video state to user's buffer
3873     <1> ; *   (buffer size: 110 bytes or 882 bytes)
3874     <1> ;
3875     <1> ;     BL > 7 : invalid (not implemented)
3876     <1> ;
3877     <1> ; 18/01/2021
3878     <1> ; SAVE/RESTORE SUPER VGA (VESA VBE 2/3) VIDEO STATE
3879     <1> ;
3880     <1> ;     BH = 14 = Save/Restore SVGA video state
3881     <1> ;
3882     <1> ;     BL = options
3883     <1> ;     bit 0 - Save (0) or Restore (1)
3884     <1> ;     bit 1 - controller hardware state
3885     <1> ;     bit 2 - BIOS data state
3886     <1> ;     bit 3 - DAC state
3887     <1> ;     bit 4 - (extended) Register state
3888     <1> ;     bit 5 - system (0) or user (1) memory
3889     <1> ;     bit 6 - verify without transfer
3890     <1> ;     bit 7 - not used (must be 0)
3891     <1> ;
3892     <1> ;     if bit 0 = 0 and bit 5 = 0
3893     <1> ;     Return: EAX = VideoStateID (>0)
3894     <1> ;     if bit 0 = 1
3895     <1> ;     ECX = VideoStateID (bit 5 = 0)
3896     <1> ;     Return: EAX = restore (transfer) size
3897     <1> ;     if bit 5 = 1
3898     <1> ;     ECX = Buffer address
3899     <1> ;     Return: EAX = transfer count (size)
3900     <1> ;
3901     <1> ;     ECX = Buffer address or VideoStateID
3902     <1> ;
3903     <1> ;     BL > 127 : invalid (not implemented)
3904     <1> ;
3905     <1> ; Note: Required buffer size may be > 2048 bytes
3906     <1> ;       (function fails when buff size > 2048 bytes)
3907     <1> ;       proper option must be used
3908     <1> ;
3909     <1> ; 18/01/2021
3910     <1> ; READ VESA EDID (EXTENDED DISPLAY IDENTIFICATION DATA)
3911     <1> ;
3912     <1> ;     BH = 15 = Read VESA EDID for connected monitor
3913     <1> ;     (copy EDID to user)
3914     <1> ;
3915     <1> ;     BL = any
3916     <1> ;
3917     <1> ; Input:
3918     <1> ;     ECX = user's (EDID) buffer address
3919     <1> ;     (buffer size: 128 bytes)
3920     <1> ; Output:
3921     <1> ;     EAX = 128 (EDID size)
3922     <1> ;     or EAX = 0 -> Error!
3923     <1> ;     (EDID not ready or buffer addr error)
3924     <1> ;
3925     <1> ;
3926     <1> ; 16/05/2016
3927 0000D3CB 31C0 xor     eax, eax
3928 0000D3CD A3[548E0100] mov    [u.r0], eax
3929 0000D3D2 20FF and     bh, bh
3930     <1> ; jnz     sysvideo_13 ; 11/07/2016
3931     <1> ; ; 23/07/2022
3932 0000D3D4 7405 jz      short sysvideo_40
3933 0000D3D6 E974020000 jmp     sysvideo_13
3934     <1> ;

```

```

3935 <1> sysvideo_40: ; 23/07/2022
3936 <1> ; 21/11/2020 (TRDOS 386 v2.0.3)
3937 <1> ; tty/text mode transfers are only for video mode 3
3938 <1>
3939 <1> ; 22/11/2020
3940 <1> ; cmp byte [CRT_MODE], 3 ; 80x25 text, 16 colors
3941 <1> ; jne sysret ; invalid (nothing to do), [u.r0] = 0
3942 <1>
3943 <1> ; 23/11/2020
3944 <1> ; bit 7,6,5,4 of BL are reserved and it must be 0
3945 <1> ; for current 'sysvideo' version
3946 <1> ; test bl, 0F0h
3947 <1> ; jnz sysret ; invalid (undefined) !
3948 <1> ; 28/01/2021
3949 <1> ; jnz short sysvideo_1_2 ; invalid (undefined) !
3950 <1> ; 28/01/2021
3951 0000D3DB 80FB07 <1> cmp bl, 7
3952 0000D3DE 776E <1> ja short sysvideo_1_2 ; invalid (undefined) !
3953 <1>
3954 <1> ; video mode 0, 80*25 text mode, CGA 16 colors
3955 <1> ; [CRT_MODE] = 3
3956 <1> ; mov bh, bl
3957 <1> ; shr bh, 2 ; 4..7 -> 1, 8..11 -> 2, 12..15 -> 3
3958 <1> ; 21/11/2020
3959 <1> ; and bh, bh
3960 <1> ; jnz sysvideo_4 ; Display page window transfer etc.
3961 <1>
3962 <1> ; 28/01/2021
3963 0000D3E0 F6C304 <1> test bl, 4 ; bit 2
3964 <1> ; jnz sysvideo_4 ; Display page window transfer
3965 <1> ; 23/07/2022
3966 0000D3E3 7405 <1> jz short sysvideo_41
3967 0000D3E5 E9A1000000 <1> jmp sysvideo_4
3968 <1> sysvideo_41: ; 23/07/2022
3969 <1> ; Display page (complete) transfer
3970 0000D3EA 80FA07 <1> cmp dl, 7
3971 <1> ; jnz sysret ; invalid (nothing to do), [u.r0] = 0
3972 0000D3ED 760A <1> jna short sysvideo_0 ; 28/01/2021
3973 0000D3EF FEC2 <1> inc dl ; 0FFh -> 0 ("use current video page")
3974 0000D3F1 755B <1> jnz short sysvideo_1_2 ; invalid
3975 <1> ; dl = 0 -> use current current page
3976 0000D3F3 8A15[CE760100] <1> mov dl, [ACTIVE_PAGE]
3977 <1> sysvideo_0:
3978 <1> ; 28/01/2021
3979 0000D3F9 80FB03 <1> cmp bl, 3
3980 0000D3FC 7206 <1> jb short sysvideo_0_0
3981 0000D3FE 09FF <1> or edi, edi
3982 0000D400 744C <1> jz short sysvideo_1_2 ; invalid
3983 0000D402 89FE <1> mov esi, edi ; save swap/exchange buffer addr
3984 <1> ; ecx = user buffer for new video page content
3985 <1> ; esi = user (swap) buffer for saving current video page
3986 <1> sysvideo_0_0:
3987 0000D404 BF00800B00 <1> mov edi, 0B8000h
3988 <1> ; dl = display page number, destination
3989 0000D409 66B80010 <1> mov ax, 4096 ; 21/11/2020
3990 0000D40D 20D2 <1> and dl, dl
3991 0000D40F 7408 <1> jz short sysvideo_1
3992 <1> ; 07/02/2021
3993 0000D411 88D6 <1> mov dh, dl
3994 <1> sysvideo_0_1:
3995 <1> ; page length = 4096 bytes (but page content is 80*25*2 bytes)
3996 0000D413 01C7 <1> add edi, eax ; 21/11/2020 ([CRT_LEN] = 1000h for mode 3)
3997 0000D415 FECE <1> dec dh
3998 0000D417 75FA <1> jnz short sysvideo_0_1
3999 <1> sysvideo_1:
4000 0000D419 80E303 <1> and bl, 3
4001 0000D41C 7535 <1> jnz short sysvideo_2 ; user to system display page transfer
4002 <1> ; system to system video page (content) transfer
4003 <1> ; cl = display page number, source
4004 0000D41E 80F907 <1> cmp cl, 7
4005 <1> ; ja sysret ; invalid (nothing to do), [u.r0] = 0
4006 0000D421 760A <1> jna short sysvideo_1_0
4007 0000D423 FEC1 <1> inc cl ; 0FFh -> 0 ("use current video page")
4008 0000D425 7527 <1> jnz short sysvideo_1_2 ; invalid
4009 0000D427 8A0D[CE760100] <1> mov cl, [ACTIVE_PAGE]
4010 <1> sysvideo_1_0:
4011 <1> ; 28/01/2021
4012 0000D42D 38D1 <1> cmp cl, dl
4013 0000D42F 741D <1> je short sysvideo_1_2 ; same video page !
4014 <1>
4015 <1> ; system to system video/display page transfer (mode 0)
4016 <1> ; 21/11/2020
4017 <1> ; mov esi, 0B8000h
4018 <1> ; movzx eax, cl
4019 <1> ; mov edx, 4096 ; [CRT_LEN] = 1000h for video mode 3
4020 <1> ; mov ecx, edx
4021 <1> ; mul edx
4022 <1> ; add esi, eax
4023 <1> ; 28/01/2021
4024 <1> ; movzx esi, cl
4025 <1> ; shl si, 12 ; * 4096
4026 <1> ; add esi, 0B8000h
4027 <1>
4028 <1> ; 28/01/2021
4029 0000D431 A3[548E0100] <1> mov [u.r0], eax ; 4096
4030 0000D436 BE00800B00 <1> mov esi, 0B8000h
4031 0000D43B 08C9 <1> or cl, cl
4032 0000D43D 7409 <1> jz short sysvideo_1_1
4033 <1> ; 07/02/2021
4034 <1> ; mov al, cl ; display/video page
4035 <1> ; xor ah, ah
4036 <1> ; shl ax, 12 ; * 4096
4037 <1> ; 23/07/2022
4038 0000D43F 30C0 <1> xor al, al
4039 <1> ; xor eax, eax
4040 0000D441 88CC <1> mov ah, cl ; CL * 256
4041 0000D443 C1E004 <1> shl eax, 4 ; * 16 = CL * 4096
4042 0000D446 01C6 <1> add esi, eax
4043 <1> sysvideo_1_1:
4044 <1> ; 21/11/2020
4045 <1> ; mov ecx, 4096
4046 <1> ; mov ecx, eax ; 4096
4047 <1> ; mov [u.r0], ecx ; 4096 bytes
4048 <1> ; 28/01/2021
4049 <1> ; mov [u.r0], cx
4050 0000D448 31C9 <1> xor ecx, ecx
4051 0000D44A B504 <1> mov ch, 4 ; mov ecx, 1024
4052 <1> ; shr cx, 2 ; / 4
4053 0000D44C F3A5 <1> rep movsd
4054 <1> sysvideo_1_2:
4055 0000D44E E92BF6FFFF <1> jmp sysret
4056 <1> sysvideo_2:
4057 0000D453 80FB02 <1> cmp bl, 2
4058 <1> ; ja sysret; invalid (user to user), [u.r0] = 0

```



```

4059          ; 28/01/2021
4060 0000D456 7226      <1>      jb      short sysvideo_3 ; user to system
4061 0000D458 7404      <1>      je      short sysvideo_2_0 ; system to user
4062          ; b1 = 3
4063 0000D45A 89CA      <1>      mov     edx, ecx ; save user's buffer addr
4064 0000D45C 89F1      <1>      mov     ecx, esi ; save swap address
4065          <1>      sysvideo_2_0:
4066          ; b1 = 2 (or b1 = 3, stage 1)
4067          ; system to user video/display page transfer (mode 0)
4068 0000D45E 89FE      <1>      mov     esi, edi
4069 0000D460 89CF      <1>      mov     edi, ecx ; user buffer ; 28/01/2021
4070          ; 21/11/2020
4071 0000D462 89C1      <1>      mov     ecx, eax ; 4096
4072 0000D464 E8A2360000 <1>      call    transfer_to_user_buffer ; fast transfer
4073          ;jc      sysret ; [u.r0] = 0
4074 0000D469 72E3      <1>      jc      short sysvideo_1_2 ; 28/01/2021
4075          ; 28/01/2021
4076 0000D46B 80FB03     <1>      cmp     b1, 3
4077 0000D46E 7408      <1>      je      short sysvideo_2_2
4078          <1>      sysvideo_2_1:
4079          ; 21/11/2020
4080 0000D470 89D0[548E0100] <1>      mov     [u.r0], ecx
4081          ; ;mov [u.r0], cx
4082          <1>      jmp     sysret
4083 0000D476 EBD6      <1>      jmp     short sysvideo_1_2
4084          <1>
4085          <1>      sysvideo_2_2:
4086          ; b1 = 3 (exchange/swap) complete display page
4087          ; esi = video page start address
4088          ; edx = user's buffer address
4089          <1>
4090          <1>      ;mov     ecx, 4096
4091 0000D478 89F7      <1>      mov     edi, esi ; video page start address
4092 0000D47A 89D6      <1>      mov     esi, edx ; user's (new page) buffer address
4093 0000D47C EB04      <1>      jmp     short sysvideo_2_3
4094          <1>      sysvideo_3:
4095          ; b1 = 1 (or b1 = 3, stage 2)
4096          ; user to system video/display page transfer (mode 0)
4097 0000D47E 89CE      <1>      mov     esi, ecx ; user buffer
4098          ; edi = video page address
4099          ; 21/11/2020
4100 0000D480 89C1      <1>      mov     ecx, eax ; 4096
4101          <1>      sysvideo_2_3:
4102 0000D482 E8CE360000 <1>      call    transfer_from_user_buffer ; fast transfer
4103          ;jc      sysret ; [u.r0] = 0
4104 0000D487 72C5      <1>      jc      short sysvideo_1_2 ; 28/01/2021
4105 0000D489 EBE5      <1>      jmp     short sysvideo_2_1
4106          <1>
4107          ; 21/11/2020
4108          <1>      ;mov     [u.r0], ecx
4109          <1>      ; ;mov [u.r0], cx
4110          <1>      ; ;jmp sysret
4111          <1>      ; jmp     short sysvideo_1_2 ; 28/01/2021
4112          <1>
4113          <1>      sysvideo_4:
4114          ; 23/07/2022
4115          ; 23/11/2020 (TRDOS 386 v2.0.3)
4116          <1>
4117          ; Display page window transfer etc.
4118 0000D48B 80E303     <1>      and     b1, 3
4119          <1>      ;jnz     sysvideo_9 ; user to system or system to user
4120          ; 23/07/2022
4121 0000D48E 7405      <1>      jz      short sysvideo_4_0
4122 0000D490 E9E3000000 <1>      jmp     sysvideo_9
4123          <1>      sysvideo_4_0:
4124          ; 21/11/2020
4125          ; system to system video/display page window transfer (mode 0)
4126 0000D495 81FE00800B00 <1>      cmp     esi, 0B8000h ; source buffer address (system)
4127          <1>      ;jb      sysret
4128          ; 23/07/2022
4129 0000D49B 7245      <1>      jb      short sysvideo_4_3 ; jmp sysret
4130 0000D49D 81FE00000C00 <1>      cmp     esi, 0B8000h+(4096*8)
4131          <1>      ; 23/07/2022
4132          <1>      ;jnb     sysret
4133 0000D4A3 733D      <1>      jnb     short sysvideo_4_3 ; jmp sysret
4134 0000D4A5 81FF00800B00 <1>      cmp     edi, 0B8000h ; destination buffer address (system)
4135          <1>      ;jb      sysret
4136          ; 23/07/2022
4137 0000D4AB 7235      <1>      jb      short sysvideo_4_3 ; jmp sysret
4138 0000D4AD 81FF00000C00 <1>      cmp     edi, 0B8000h+(4096*8)
4139          <1>      ; 23/07/2022
4140          <1>      ;jnb     sysret
4141 0000D4B3 732D      <1>      jnb     short sysvideo_4_3 ; jmp sysret
4142          <1>
4143 0000D4B5 51          <1>      push    ecx ; x1 and y1 position - top left column, row
4144 0000D4B6 0FB7C1     <1>      movzx   eax, cx ; top left column
4145          <1>      ; 21/11/2020
4146 0000D4B9 C1E910     <1>      shr     ecx, 16 ; top row
4147 0000D4BC 740E      <1>      jz      short sysvideo_4_1 ; bypass following code
4148 0000D4BE 52          <1>      push    edx ; x2 and y2 position - bottom right column, row
4149 0000D4BF 50          <1>      push    eax
4150 0000D4C0 66B8A000     <1>      mov     ax, 80*2 ; 80 columns, 160 bytes per row
4151 0000D4C4 F7E1      <1>      mul     ecx
4152          <1>      ; eax = offset for start row number
4153 0000D4C6 01C6      <1>      add     esi, eax
4154 0000D4C8 01C7      <1>      add     edi, eax
4155 0000D4CA 58          <1>      pop     eax
4156 0000D4CB 5A          <1>      pop     edx
4157          <1>      sysvideo_4_1:
4158          ; shl     ax, 1 ; * 2 ; convert start column number to offset
4159          ; 23/07/2022
4160 0000D4CC D1E0      <1>      shl     eax, 1
4161 0000D4CE 7404      <1>      jz      short sysvideo_4_2
4162 0000D4D0 01C6      <1>      add     esi, eax
4163 0000D4D2 01C7      <1>      add     edi, eax
4164          <1>      ; esi = source page window start offset
4165          <1>      ; edi = destination page window start offset
4166          <1>      sysvideo_4_2:
4167 0000D4D4 59          <1>      pop     ecx
4168          <1>      ;mov     eax, 0B8000h+(80*25*2*8)
4169          <1>      ; 21/11/2020
4170 0000D4D5 B800000C00 <1>      mov     eax, 0B8000h+(4096*8)
4171 0000D4DA 39C6      <1>      cmp     esi, eax
4172          <1>      ;jnb     sysret ; out of video page
4173          ; 23/07/2022
4174 0000D4DC 7304      <1>      jnb     short sysvideo_4_3 ; jmp sysret
4175 0000D4DE 39C6      <1>      cmp     esi, eax
4176          <1>      ;jnb     sysret ;out of video page
4177          ; 23/07/2022
4178 0000D4E0 7205      <1>      jb      short sysvideo_4_4
4179          <1>      sysvideo_4_3:
4180 0000D4E2 E997F5FFFF     <1>      jmp     sysret
4181          <1>      sysvideo_4_4:
4182 0000D4E7 56          <1>      push    esi ; ****

```

```

4183 0000D4E8 57      <1>      push    edi ; ***
4184 0000D4E9 52      <1>      push    edx ; **
4185 0000D4EA 51      <1>      push    ecx ; *
4186 0000D4EB C1E910  <1>      shr     ecx, 16 ; top row
4187 0000D4EE C1EA10  <1>      shr     edx, 16 ; bottom row
4188                <1>      ; 21/11/2020
4189                <1>      ;cmp    ecx, 24 ; max. 25 rows
4190 0000D4F1 6683F918 <1>      cmp     cx, 24
4191 0000D4F5 7778      <1>      ja      short sysvideo_6 ; invalid, [u.r0] = 0
4192                <1>      ;cmp    edx, 24 ; max. 25 rows
4193 0000D4F7 6683FA18 <1>      cmp     dx, 24
4194 0000D4FB 7772      <1>      ja      short sysvideo_6 ; invalid, [u.r0] = 0
4195 0000D4FD 28CA      <1>      sub     dl, cl ; end >= start
4196 0000D4FF 726E      <1>      jc      short sysvideo_6 ; invalid, [u.r0] = 0
4197                <1>      ; 21/11/2020
4198 0000D501 89D3      <1>      mov     ebx, edx ; row count - 1
4199 0000D503 7415      <1>      jz      short sysvideo_4_5
4200 0000D505 50          <1>      push    eax ; *****
4201 0000D506 B8A0000000 <1>      mov     eax, 80*2 ; bytes per row
4202 0000D50B F7E3      <1>      mul     ebx ; 21/11/2020
4203                <1>      ; eax = window end offset
4204                <1>      ; (for the last row, before adding column bytes)
4205 0000D50D 01C6      <1>      add     esi, eax
4206 0000D50F 01C7      <1>      add     edi, eax
4207 0000D511 58          <1>      pop     eax ; ***** ; mode 3 video memory end (0C000h)
4208 0000D512 39C6      <1>      cmp     esi, eax
4209                <1>      ;ja      short sysvideo_6 ; invalid, [u.r0] = 0
4210 0000D514 7359      <1>      jnb     short sysvideo_6 ; 21/11/2020
4211 0000D516 39C7      <1>      cmp     edi, eax
4212                <1>      ;ja      short sysvideo_6 ; invalid, [u.r0] = 0
4213 0000D518 7355      <1>      jnb     short sysvideo_6 ; 21/11/2020
4214                <1>      sysvideo_4_5:
4215 0000D51A 59          <1>      pop     ecx ; *
4216 0000D51B 5A          <1>      pop     edx ; **
4217 0000D51C 81E1FFFF0000 <1>      and     ecx, 0FFFFh
4218 0000D522 81E2FFFF0000 <1>      and     edx, 0FFFFh
4219                <1>      ; 21/11/2020
4220                <1>      ;cmp    ecx, 79 ; max. 80 columns
4221 0000D528 6683F94F <1>      cmp     cx, 79
4222 0000D52C 7743      <1>      ja      short sysvideo_7 ; invalid, [u.r0] = 0
4223                <1>      ;cmp    edx, 79 ; max. 80 columns
4224 0000D52E 6683FA4F <1>      cmp     dx, 79
4225 0000D532 773D      <1>      ja      short sysvideo_7 ; invalid, [u.r0] = 0
4226 0000D534 28CA      <1>      sub     dl, cl
4227 0000D536 7639      <1>      jna     short sysvideo_7 ; invalid, [u.r0] = 0
4228                <1>      ; 21/11/2020
4229 0000D538 740E      <1>      jz      short sysvideo_4_6
4230                <1>      ; edx = column count (width) - 1
4231 0000D53A D0E2      <1>      shl     dl, 1 ; * 2 ; byte offset (in end row)
4232 0000D53C 01D6      <1>      add     esi, edx
4233 0000D53E 01D7      <1>      add     edi, edx
4234                <1>      ; esi = source page window end offset
4235                <1>      ; edi = destination page window end offset
4236 0000D540 39C6      <1>      cmp     esi, eax ; video memory end
4237                <1>      ;ja      short sysvideo_7
4238 0000D542 732D      <1>      jnb     short sysvideo_7 ; 21/11/2020
4239 0000D544 39C7      <1>      cmp     edi, eax ; video memory end
4240                <1>      ;ja      short sysvideo_7
4241 0000D546 7329      <1>      jnb     short sysvideo_7 ; 21/11/2020
4242                <1>      sysvideo_4_6:
4243 0000D548 5F          <1>      pop     edi ; ***
4244 0000D549 5E          <1>      pop     esi ; ****
4245 0000D54A FEC3      <1>      inc     bl ; row count - 1 -> row count
4246 0000D54C FEC2      <1>      inc     dl ; column count
4247 0000D54E 88D7      <1>      mov     bh, dl
4248 0000D550 D0E2      <1>      shl     dl, 1 ; convert column count to byte offset
4249                <1>      ; 21/11/2020
4250                <1>      ; esi = source page window start offset
4251                <1>      ; edi = destination page window start offset
4252 0000D552 B8A0000000 <1>      mov     eax, 80*2 ; bytes per row
4253                <1>      ; Note: 160 bytes per row (even if move count < 160)
4254                <1>      sysvideo_5:
4255 0000D557 88F9      <1>      mov     cl, bh ; move/transfer -word- count per row
4256 0000D559 0115[548E0100] <1>      add     [u.r0], edx ; transfer count in bytes
4257 0000D55F F366A5 <1>      rep     movsw
4258 0000D562 01C6      <1>      add     esi, eax ; + 160 bytes to next row
4259 0000D564 01C7      <1>      add     edi, eax ; + 160 bytes to next row
4260 0000D566 FECB      <1>      dec     bl ; remain count of rows
4261 0000D568 75ED      <1>      jnz     short sysvideo_5
4262 0000D56A E90FF5FFFF <1>      jmp     sysret
4263                <1>
4264                <1>      sysvideo_6:
4265 0000D56F 59          <1>      pop     ecx ; *
4266 0000D570 5A          <1>      pop     edx ; **
4267                <1>      sysvideo_7:
4268 0000D571 5F          <1>      pop     edi ; ***
4269 0000D572 5E          <1>      pop     esi ; ****
4270                <1>      sysvideo_8:
4271 0000D573 E906F5FFFF <1>      jmp     sysret
4272                <1>
4273                <1>      sysvideo_9:
4274                <1>      ; user to system or system to user window transfer
4275                <1>      ; 28/01/2021 (b1 = 3 -> swap/exchange)
4276                <1>      ;cmp    b1, 2
4277                <1>      ;ja      sysret ; user to user transfer is invalid
4278                <1>      ; [u.r0] = 0
4279                <1>      ;ja      short sysvideo_8 ; 26/12/2020
4280                <1>
4281                <1>      ; 28/01/2021
4282 0000D578 80FB02 <1>      cmp     b1, 2
4283 0000D57B 7604      <1>      jna     short sysvideo_9_8
4284                <1>
4285                <1>      ; swap/ exchange video memory and user mem windows
4286                <1>      ; edi = swap address in user's memory space
4287 0000D57D 21FF      <1>      and     edi, edi
4288 0000D57F 74F2      <1>      jz      short sysvideo_8 ; invalid ; 28/01/2021
4289                <1>
4290                <1>      sysvideo_9_8:
4291 0000D581 56          <1>      push    esi ; ****
4292 0000D582 57          <1>      push    edi ; ***
4293 0000D583 52          <1>      push    edx ; **
4294 0000D584 51          <1>      push    ecx ; *
4295                <1>
4296 0000D585 C1E910 <1>      shr     ecx, 16 ; top row
4297 0000D588 C1EA10 <1>      shr     edx, 16 ; bottom row
4298                <1>
4299                <1>      ; 21/11/2020
4300                <1>      ;cmp    ecx, 24 ; max. 25 rows
4301 0000D58B 6683F918 <1>      cmp     cx, 24
4302 0000D58F 77DE      <1>      ja      short sysvideo_6 ; invalid, [u.r0] = 0
4303                <1>      ;cmp    edx, 24 ; max. 25 rows
4304 0000D591 6683FA18 <1>      cmp     dx, 24
4305 0000D595 77D8      <1>      ja      short sysvideo_6 ; invalid, [u.r0] = 0
4306 0000D597 28CA      <1>      sub     dl, cl

```

```

4307 0000D599 72D4      <1>      jc      short sysvideo_6 ; invalid, [u.r0] = 0
4308                  <1>
4309                  <1>      ;mov     ch, cl ; top row
4310                  <1>      ;mov     cl, [ACTIVE_PAGE]
4311                  <1>
4312                  <1>      ;mov     edi, 80*25*2 ; 4000
4313                  <1>      ; 21/11/2020
4314                  <1>      ;mov     edi, 4096 ; [CRT_LEN = 4096 for video mode 3
4315                  <1>      ;shl     edi, cl ; ! wrong for page 2 to page 7 !
4316                  <1>      ;add     edi, 0B8000h - 80*25*2
4317                  <1>      ;add     edi, 0B8000h - 1000h ; - 4096
4318                  <1>
4319                  <1>      ; 21/11/2020
4320                  <1>      ;xor     eax, eax
4321                  <1>      ;mov     edi, 0B8000h
4322                  <1>      ;and     cl, cl ; is video page = 0 ?
4323                  <1>      ;jz      short sysvideo_9_1 ; yes
4324                  <1>      ; eax = 0
4325                  <1>
4326                  <1> ;sysvideo_9_0:
4327                  <1>      ;add     ax, 4096
4328                  <1>      ;dec     cl
4329                  <1>      ;jnz     short sysvideo_9_0
4330                  <1>      ;add     edi, eax
4331                  <1>      ;      ; edi = video page start address
4332                  <1>
4333                  <1>      ; 21/11/2020
4334 0000D59B BF00800B00  <1>      mov     edi, 0B8000h
4335 0000D5A0 803D[CE760100]00 <1>      cmp     byte [ACTIVE_PAGE], 0
4336 0000D5A7 760C      <1>      jna     short sysvideo_9_1
4337                  <1> stsvideo_9_0:
4338 0000D5A9 B010      <1>      mov     al, 16 ; 4096/256
4339 0000D5AB F625[CE760100] <1>      mul     byte [ACTIVE_PAGE]
4340 0000D5B1 86C4      <1>      xchg     ah, al ; * 256
4341 0000D5B3 01C7      <1>      add     edi, eax
4342                  <1>      ; edi = video page start address
4343                  <1> sysvideo_9_1:
4344                  <1>      ; bl = transfer direction
4345                  <1>      ;      (1 = from user, 2 = to user)
4346                  <1>      ;      (3 = swap) ; 28/01/2021
4347 0000D5B5 88D7      <1>      mov     bh, dl ; row count - 1
4348                  <1>      ;mov     dl, ch ; top row
4349                  <1>      ; 21/11/2020
4350 0000D5B7 08C9      <1>      or      cl, cl ; top row number
4351 0000D5B9 7408      <1>      jz      short sysvideo_9_2
4352                  <1>
4353                  <1>      ;mov     eax, 80*2
4354 0000D5BB 66B8A000 <1>      mov     ax, 80*2 ; 160, bytes per row
4355 0000D5BF F7E1      <1>      mul     ecx ; 22/11/2020
4356 0000D5C1 01C7      <1>      add     edi, eax
4357                  <1>      ; edi = window start address for top row
4358                  <1> sysvideo_9_2:
4359 0000D5C3 59      <1>      pop     ecx ; *
4360 0000D5C4 5A      <1>      pop     edx ; **
4361 0000D5C5 81E1FFFF0000 <1>      and     ecx, 0FFFFh
4362 0000D5CB 81E2FFFF0000 <1>      and     edx, 0FFFFh
4363                  <1>      ; 21/11/2020
4364                  <1>      ;cmp     ecx, 79 ; max. 80 columns
4365 0000D5D1 6683F94F <1>      cmp     cx, 79
4366 0000D5D5 779A      <1>      ja     short sysvideo_7 ; invalid, [u.r0] = 0
4367                  <1>      ;cmp     edx, 79 ; max. 80 columns
4368 0000D5D7 6683FA4F <1>      cmp     dx, 79
4369 0000D5DB 7794      <1>      ja     short sysvideo_7 ; invalid, [u.r0] = 0
4370                  <1>
4371 0000D5DD 28CA      <1>      sub     dl, cl
4372 0000D5DF 7290      <1>      jc     short sysvideo_7 ; invalid, [u.r0] = 0
4373                  <1>
4374 0000D5E1 08C9      <1>      or      cl, cl ; left column
4375 0000D5E3 7404      <1>      jz     short sysvideo_9_3 ; 0
4376                  <1>
4377                  <1>      ; 21/11/2020
4378 0000D5E5 D0E1      <1>      shl     cl, 1 ; column * 2
4379 0000D5E7 01CF      <1>      add     edi, ecx
4380                  <1>      ; edi = window start addr for top left column
4381                  <1> sysvideo_9_3:
4382 0000D5E9 88D1      <1>      mov     cl, dl
4383 0000D5EB FEC1      <1>      inc     cl ; column count
4384 0000D5ED D0E1      <1>      shl     cl, 1 ; column count * 2
4385                  <1>      ; ecx = transfer count per row
4386                  <1>
4387 0000D5EF 58      <1>      pop     eax ; *** (swap address)
4388 0000D5F0 5E      <1>      pop     esi ; ****
4389                  <1>
4390 0000D5F1 FEC7      <1>      inc     bh ; row count
4391                  <1>
4392                  <1>      ;mov     edx, 80*2
4393 0000D5F3 B2A0      <1>      mov     dl, 80*2 ; bytes per row
4394                  <1>
4395                  <1>      ;cmp     bl, 1 ; transfer direction
4396                  <1>      ;ja     short sysvideo_11 ; system to user transfer
4397                  <1>      ; 28/01/2021
4398 0000D5F5 F6C301 <1>      test    bl, 1
4399 0000D5F8 7439      <1>      jz     short sysvideo_11 ; system to user transfer
4400                  <1>
4401                  <1>      ; user to system video/display page window transfer (mode 0)
4402 0000D5FA 21C0      <1>      and     eax, eax ; swap address
4403 0000D5FC 741B      <1>      jz     short sysvideo_10 ; no window swap
4404                  <1> sysvideo_9_7: ; 28/01/2021
4405                  <1>      ; save previous window content in user's buffer (swap address)
4406 0000D5FE 56      <1>      push    esi ; user buffer
4407 0000D5FF 57      <1>      push    edi ; beginning address of the window
4408                  <1>      ; 21/11/2020
4409 0000D600 53      <1>      push    ebx ; save bh
4410 0000D601 89FE      <1>      mov     esi, edi
4411 0000D603 89C7      <1>      mov     edi, eax
4412                  <1> sysvideo_9_4:
4413 0000D605 E801350000 <1>      call    transfer_to_user_buffer ; fast transfer
4414 0000D60A 7208      <1>      jc     short sysvideo_9_5
4415                  <1>      ; ecx = actual transfer count (must be same with input)
4416 0000D60C 01D6      <1>      add     esi, edx ; next row address of (video page) window
4417 0000D60E 01CF      <1>      add     edi, ecx ; next row address of user's window
4418                  <1>      ; Note: ecx may be less than row length of video page
4419                  <1>      ; user's window uses offset according to window width
4420 0000D610 FECF      <1>      dec     bh
4421 0000D612 75F1      <1>      jnz     short sysvideo_9_4 ; repeat for next row
4422                  <1> sysvideo_9_5:
4423 0000D614 5B      <1>      pop     ebx ; restore bh
4424 0000D615 5F      <1>      pop     edi
4425 0000D616 5E      <1>      pop     esi
4426                  <1>      ;jnc     short sysvideo_10
4427 0000D617 7215      <1>      jc     short sysvideo_9_6 ; 28/01/2021
4428                  <1> ;sysvideo_9_6:
4429                  <1>      ; jmp     sysret ; [u.r0] = 0
4430                  <1>

```

```

4431 <1> sysvideo_10:
4432 <1> ; user to system video/display page window transfer (mode 0)
4433 <1> ; esi = user buffer
4434 0000D619 E837350000 <1> call transfer_from_user_buffer ; fast transfer
4435 <1> ;jc sysret
4436 0000D61E 720E <1> jc short sysvideo_9_6 ; 28/01/2021
4437 <1> ; ecx = actual transfer count (must be same with input)
4438 0000D620 010D[548E0100] <1> add [u.r0], ecx ; actual transfer count
4439 0000D626 01D7 <1> add edi, edx ; next row address of (video page) window
4440 0000D628 01CE <1> add esi, ecx ; next row address of user's window
4441 <1> ; Note: ecx may be less than row length of video page
4442 <1> ; user's window uses offset according to window width
4443 0000D62A FECF <1> dec bh
4444 0000D62C 75EB <1> jnz short sysvideo_10 ; repeat for next row
4445 <1> ;jmp sysret
4446 <1> sysvideo_9_6:
4447 0000D62E E94BF4FFFF <1> jmp sysret
4448 <1>
4449 <1> sysvideo_11:
4450 <1> ; system to user video/display page window transfer (mode 0)
4451 0000D633 87F7 <1> xchg edi, esi
4452 <1> sysvideo_12:
4453 <1> ; esi = beginning addr of the (screen, video page) window
4454 <1> ; edi = user's buffer
4455 0000D635 E8D1340000 <1> call transfer_to_user_buffer ; fast transfer
4456 <1> ;jc sysret
4457 <1> ; 23/07/2022
4458 0000D63A 72F2 <1> jc short sysvideo_9_6 ; jmp sysret
4459 <1>
4460 <1> ; ecx = actual transfer count (must be same with input)
4461 0000D63C 010D[548E0100] <1> add [u.r0], ecx
4462 0000D642 01D6 <1> add esi, edx ; next row (edx = 160)
4463 0000D644 01CF <1> add edi, ecx ; next row of the user's window
4464 <1> ; (ecx <= 160)
4465 0000D646 FECF <1> dec bh
4466 0000D648 75EB <1> jnz short sysvideo_12
4467 <1> sysvideo_12_0:
4468 0000D64A E92FF4FFFF <1> jmp sysret
4469 <1>
4470 <1> sysvideo_13:
4471 <1> ; 28/12/2020
4472 0000D64F 80FF01 <1> cmp bh, 1
4473 0000D652 7752 <1> ja short sysvideo_15 ; 23/11/2020
4474 <1>
4475 <1> ; 25/02/2021
4476 <1> ; 12/02/2021
4477 <1> ; 29/01/2021, 31/01/2021
4478 <1> ; 23/11/2020 (TRDOS 386 v2.0.3)
4479 <1> ; (major modification, from mode 13h to all VGA modes,
4480 <1> ; except super VGA modes and liner frame buffer method)
4481 <1>
4482 <1> ; BH = 1 = VGA Graphics mode (0A0000h) data transfers
4483 <1>
4484 <1> ; 29/01/2021
4485 0000D654 66B84001 <1> mov ax, 320 ; 320 pixels
4486 0000D658 F6C380 <1> test bl, 80h ; bit 7 (screen width, 640 pixels)
4487 0000D65B 7407 <1> jz short sysvideo_13_0
4488 <1> ;shl ax, 1 ; 640 pixels
4489 <1> ; 23/07/2022
4490 0000D65D D1E0 <1> shl eax, 1
4491 <1>
4492 0000D65F 80E37F <1> and bl, 7Fh
4493 0000D662 7405 <1> jz short sysvideo_14
4494 <1> sysvideo_13_0:
4495 <1> ; 29/01/2021
4496 0000D664 80FB41 <1> cmp bl, 41h
4497 0000D667 77E1 <1> ja short sysvideo_12_0 ; invalid (unknown) sub function
4498 <1> sysvideo_14:
4499 0000D669 66A3[769D0100] <1> mov [v_width], ax ; save screen width
4500 0000D66F C705[7A9D0100]0000- <1> mov dword [v_mem], 0A0000h ; save video memory address
4500 0000D677 0A00 <1>
4501 0000D679 C705[7E9D0100]0000- <1> mov dword [v_siz], 65536 ; save video memory size
4501 0000D681 0100 <1>
4502 0000D683 C705[869D0100]0000- <1> mov dword [v_end], 0B0000h ; save end of video page
4502 0000D68B 0B00 <1>
4503 0000D68D B708 <1> mov bh, 8
4504 0000D68F 883D[799D0100] <1> mov [v_bpp], bh ; 8 ; bits per pixel (256 colors)
4505 0000D695 881D[789D0100] <1> mov [v_ops], bl ; VGA data transfer options
4506 <1> ;mov [maskbuff], edi ; 25/02/2021
4507 0000D69B 893D[8A9D0100] <1> mov [maskcolor], edi ; 25/02/2021
4508 <1> ; save mask color or bitmask buffer address
4509 0000D6A1 E9BE000000 <1> jmp sysvideo_15_7
4510 <1>
4511 <1> sysvideo_15:
4512 <1> ; 23/07/2022
4513 <1> ; 28/12/2020
4514 0000D6A6 80FF02 <1> cmp bh, 2
4515 <1> ;ja sysvideo_16
4516 <1> ; 23/07/2022
4517 0000D6A9 7605 <1> jna short sysvideo_15_17
4518 0000D6AB E9ED1E0000 <1> jmp sysvideo_16
4519 <1> sysvideo_15_17: ; 23/07/2022
4520 <1> ; 25/02/2021
4521 <1> ; 12/02/2021
4522 <1> ; 30/01/2021 - 31/01/2021
4523 <1> ; 01/01/2021 - 29/01/2021
4524 <1> ; 26/12/2020 - 27/12/2020
4525 <1> ; 25/12/2020 (TRDOS 386 v2.0.3)
4526 <1> ;
4527 <1> ; BH = 2 = SVGA (VESA VBE) Graphics mode (LFB) data transfers
4528 <1>
4529 <1> ; 25/12/2020
4530 <1> ; resolution table entry will be saved into EBP register
4531 <1>
4532 0000D6B0 803D[86090000]02 <1> cmp byte [vbe3], 2 ; VESA VBE 3 video bios
4533 <1> ; or BOCHS/QEMU/VIRTUALBOX emu video bios
4534 0000D6B7 724E <1> jb short sysvideo_15_4 ; no, nothing to do !
4535 0000D6B9 770B <1> ja short sysvideo_15_0 ; yes
4536 <1>
4537 <1> ; Only Bochs/Plex86 (emu) vbe2 video bios is usable in pmid
4538 <1> ; (if [vbe3] = 2)
4539 0000D6BB A0[87090000] <1> mov al, [vbe2bios] ; Bochs vbios sign is from C0h to C5h
4540 0000D6C0 24F0 <1> and al, 0F0h
4541 0000D6C2 3CC0 <1> cmp al, 0C0h
4542 0000D6C4 7541 <1> jne short sysvideo_15_4 ; unknown (vbe2) video bios
4543 <1> sysvideo_15_0:
4544 <1> ; 29/01/2021
4545 0000D6C6 80FB41 <1> cmp bl, 41h
4546 0000D6C9 773C <1> ja short sysvideo_15_4 ; invalid (unknown) sub function
4547 <1> ; 29/01/2021
4548 0000D6CB 881D[789D0100] <1> mov [v_ops], bl ; SVGA data transfer options
4549 <1>
4550 0000D6D1 89D8 <1> mov eax, ebx ; hw of ebx is vesa vbe video mode
4551 0000D6D3 C1E810 <1> shr eax, 16 ; ax = vesa vbe video mode

```

```

4552 0000D6D6 7513      <1>      jnz      short sysvideo_15_2
4553                    <1>      ; ax = 0
4554                    <1>
4555                    <1>      ; check & use current video mode
4556 0000D6D8 803D[1E680000]FF <1>      cmp      byte [CRT_MODE], 0FFh ; extended (SVGA) mode ?
4557 0000D6DF 7526      <1>      jne      short sysvideo_15_4 ; no
4558                    <1> sysvideo_15_1:
4559                    <1>      ; use current vbe (svga) video mode
4560 0000D6E1 66A1[0E9D0100] <1>      mov      ax, [video_mode] ; extended (SVGA, VESA VBE) mode
4561 0000D6E7 6625FF01    <1>      and      ax, 1FFh ; vesa vbe video mode: 1XXh
4562                    <1> sysvideo_15_2:
4563                    <1>      ; 29/01/2021
4564                    <1>      ;mov      [maskbuff], edi ; 25/02/2021
4565 0000D6EB 893D[8A9D0100] <1>      mov      [maskcolor], edi ; 25/02/2021
4566                    <1>      ; save mask color or bitmask buffer address
4567 0000D6F1 BD[AA6B0000] <1>      mov      ebp, b_vbe_modes ; vbe mode table (in 'vidata.s')
4568                    <1> sysvideo_15_3:
4569 0000D6F6 663B4500    <1>      cmp      ax, [ebp]
4570 0000D6FA 7410      <1>      je       short sysvideo_15_5
4571 0000D6FC 83C508      <1>      add      ebp, 8 ; vbe mode table entry size
4572 0000D6FF 81FD[6A6C0000] <1>      cmp      ebp, end_of_b_vbe_modes
4573 0000D705 72EF      <1>      jb       short sysvideo_15_3
4574                    <1> sysvideo_15_4:
4575                    <1>      ; desired video mode is not a valid (implemented)
4576                    <1>      ; extended (VESA VBE, SVGA) video mode
4577                    <1>      ;
4578                    <1>      ; nothing to do !
4579                    <1>
4580                    <1>      ; [u.r0] = 0 ; return value of EAX
4581 0000D707 E972F3FFFF <1>      jmp      sysret
4582                    <1>
4583                    <1> sysvideo_15_5:
4584                    <1>      ; get LFB address
4585 0000D70C A1[1C9D0100] <1>      mov      eax, [LFB_ADDR] ; [LFB_Info+LFBINFO.LFB_addr]
4586 0000D711 09C0      <1>      or       eax, eax
4587 0000D713 7509      <1>      jnz      short sysvideo_15_6
4588 0000D715 66A1[1A0F0000] <1>      mov      ax, [def_LFB_addr] ; default LFB addr
4589                    <1>      ; (for vbe mode 118h)
4590 0000D71B C1E010      <1>      shl      eax, 16
4591                    <1>      ; 27/12/2020
4592                    <1>      ;jz       short sysvideo_15_4
4593                    <1> sysvideo_15_6:
4594                    <1>      ; 29/01/2021
4595 0000D71E A3[7A9D0100] <1>      mov      [v_mem], eax ; save video memory address
4596                    <1>
4597                    <1>      ; 27/12/2020
4598                    <1>      ; 26/12/2020
4599 0000D723 8B4502      <1>      mov      eax, [ebp+2] ; width, height
4600                    <1>      ; 29/01/2021
4601 0000D726 66A3[769D0100] <1>      mov      [v_width], ax ; save screen width
4602                    <1>      ; 28/12/2020
4603 0000D72C 8A7D06      <1>      mov      bh, [ebp+6] ; bpp
4604                    <1>      ; 28/02/2021
4605                    <1>      ; check default truecolor bpp value and use
4606                    <1>      ; 32bpp instead of 24bpp if the default value
4607                    <1>      ; has been set to 32bpp.
4608 0000D72F 80FF18      <1>      cmp      bh, 24
4609 0000D732 750B      <1>      jne      short sysvideo_15_16
4610 0000D734 803D[63730100]20 <1>      cmp      byte [truecolor], 32
4611                    <1>      ; Default truecolor bpp value,
4612                    <1>      ; it is 32 for VBE3 video bios
4613                    <1>      ; (it can be set to 32 or 24)
4614 0000D73B 7502      <1>      jne      short sysvideo_15_16 ; not VBE3 !
4615                    <1>      ; or it is set to 24
4616 0000D73D B720      <1>      mov      bh, 32
4617                    <1>      ; 28/02/2021
4618                    <1> sysvideo_15_16:
4619                    <1>      ; 29/01/2021
4620 0000D73F 883D[799D0100] <1>      mov      [v_bpp], bh ; bits per pixel
4621                    <1>
4622 0000D745 52          <1>      push     edx ; *
4623 0000D746 0FB7D0      <1>      movzx    edx, ax ; width
4624 0000D749 C1E810      <1>      shr      eax, 16 ; height
4625 0000D74C F7E2      <1>      mul      edx
4626                    <1>      ; eax = linear frame buffer size (pixels)
4627                    <1>      ; 29/01/2021
4628 0000D74E A3[7E9D0100] <1>      mov      [v_siz], eax ; save video page size
4629 0000D753 E8FD000000 <1>      call     pixels_to_byte_count
4630 0000D758 0305[7A9D0100] <1>      add      eax, [v_mem]
4631 0000D75E A3[869D0100] <1>      mov      [v_end], eax ; save end of video page
4632 0000D763 5A          <1>      pop      edx ; *
4633                    <1>
4634                    <1>      ; bh = bits per pixel
4635                    <1>      ; (bh will not be used after here, 29/01/2021)
4636                    <1>
4637                    <1>      ; b1 = pixel operations & options
4638                    <1>      ; ecx, edx, esi, edi input parameters
4639                    <1>      ; [maskcolor] = edi input ; 25/02/2021
4640                    <1>
4641                    <1> sysvideo_15_7:
4642                    <1>      ; 29/01/2021
4643                    <1>      ;test     byte [v_ops], 40h ; system to user ?
4644 0000D764 F6C340      <1>      test     bl, 40h
4645 0000D767 7517      <1>      jnz      short sysvideo_15_9
4646                    <1>
4647 0000D769 31C0      <1>      xor      eax, eax
4648 0000D76B 88D8      <1>      mov      al, bl
4649 0000D76D B8[93D80000] <1>      mov      ebx, pixel_ops
4650 0000D772 240F      <1>      and      al, 0Fh ; isolate 16 pixel operations
4651 0000D774 C0E002      <1>      shl      al, 2 ; * 4 for dword table pointers
4652 0000D777 01C3      <1>      add      ebx, eax
4653                    <1>
4654                    <1>      ; ebx = subroutine address
4655                    <1>
4656                    <1>      ; ecx, edx, esi, edi input parameters
4657                    <1>      ; [maskbuff] = edi input
4658                    <1>      ; [maskcolor] = edi input ; 25/02/2021
4659                    <1>
4660 0000D779 FF13      <1>      call     [ebx]
4661                    <1> sysvideo_15_8:
4662 0000D77B E9FEF2FFFF <1>      jmp      sysret
4663                    <1>
4664                    <1> sysvideo_15_9:
4665                    <1>      ; system to user display page or window copy
4666                    <1>      ;test     byte [v_ops], 1 ; window copy ?
4667 0000D780 F6C301      <1>      test     bl, 1
4668 0000D783 7521      <1>      jnz      short sysvideo_15_10
4669                    <1>
4670                    <1>      ; display page (full screen copy)
4671 0000D785 8B35[7A9D0100] <1>      mov      esi, [v_mem] ; LFB start address
4672 0000D78B A1[7E9D0100] <1>      mov      eax, [v_siz]
4673 0000D790 E8C0000000 <1>      call     pixels_to_byte_count
4674 0000D795 89C1      <1>      mov      ecx, eax ; transfer count in bytes
4675                    <1>      ;edi = user's buffer address

```

```

4676 0000D797 E86F330000 <1> call transfer_to_user_buffer
4677 0000D79C 72DD <1> jc short sysvideo_15_8
4678 0000D79E 890D[548E0100] <1> mov [u.r0], ecx
4679 0000D7A4 EBD5 <1> jmp short sysvideo_15_8
4680 <1>
4681 <1> sysvideo_15_10:
4682 0000D7A6 E820000000 <1> call sysvideo_15_12 ; window preparations
4683 0000D7AB 72CE <1> jc short sysvideo_15_8
4684 <1>
4685 0000D7AD 8B35[829D0100] <1> mov esi, [v_str]
4686 <1> sysvideo_15_11:
4687 <1> ; esi = window's current row address (video mem)
4688 <1> ; edi = current row (virtual) addr in user's buff
4689 <1> ; ecx = transfer count per row
4690 0000D7B3 E853330000 <1> call transfer_to_user_buffer
4691 0000D7B8 72C1 <1> jc short sysvideo_15_8
4692 0000D7BA 010D[548E0100] <1> add [u.r0], ecx
4693 0000D7C0 4B <1> dec ebx
4694 0000D7C1 74B8 <1> jz short sysvideo_15_8 ; ok.
4695 <1> ; next row
4696 0000D7C3 01CF <1> add edi, ecx ; next row in user's buffer
4697 0000D7C5 01D6 <1> add esi, edx ; next row of window (system)
4698 0000D7C7 EBEA <1> jmp short sysvideo_15_11
4699 <1>
4700 <1> sysvideo_15_14:
4701 0000D7C9 F9 <1> stc ; error !
4702 <1> sysvideo_15_15:
4703 0000D7CA C3 <1> retn
4704 <1>
4705 <1> sysvideo_15_12:
4706 <1> ; 30/01/2021
4707 <1> ; 29/01/2021
4708 <1> ; window address preparations for window copy
4709 0000D7CB 6621D2 <1> and dx, dx
4710 0000D7CE 74F9 <1> jz short sysvideo_15_14 ; invalid (zero columns)
4711 <1> ; test
4712 <1> ; jz short sysvideo_15_14 ; invalid (zero rows)
4713 0000D7D0 81FA00000100 <1> cmp edx, 65536
4714 0000D7D6 72F2 <1> jb short sysvideo_15_15 ; invalid (zero rows)
4715 0000D7D8 89C8 <1> mov eax, ecx ; start position (row, column)
4716 0000D7DA E899000000 <1> call calc_pixel_offset
4717 0000D7DF 3B05[7E9D0100] <1> cmp eax, [v_siz]
4718 0000D7E5 73E2 <1> jnb short sysvideo_15_14 ; out of display page
4719 <1> ; nothing to do
4720 0000D7E7 E869000000 <1> call pixels_to_byte_count
4721 0000D7EC 0305[7A9D0100] <1> add eax, [v_mem]
4722 0000D7F2 A3[829D0100] <1> mov [v_str], eax ; window start address
4723 <1> ; (addr of top left corner)
4724 <1> ; check column limit
4725 0000D7F7 89C8 <1> mov eax, ecx
4726 0000D7F9 6601D0 <1> add ax, dx ; add columns to start column
4727 0000D7FC 72CC <1> jc short sysvideo_15_15 ; cf = 1
4728 0000D7FE 663B05[769D0100] <1> cmp ax, [v_width]
4729 0000D805 77C2 <1> ja short sysvideo_15_14
4730 <1>
4731 0000D807 89D0 <1> mov eax, edx ; size
4732 0000D809 2B00000100 <1> sub eax, 65536 ; row count -> 0 based row #
4733 0000D80E E865000000 <1> call calc_pixel_offset
4734 0000D813 3B05[7E9D0100] <1> cmp eax, [v_siz] ; video (display) page size
4735 0000D819 77AE <1> ja short sysvideo_15_14 ; out of display page
4736 <1> ; nothing to do
4737 0000D81B E835000000 <1> call pixels_to_byte_count
4738 0000D820 0305[829D0100] <1> add eax, [v_str] ; window start address
4739 0000D826 3B05[869D0100] <1> cmp eax, [v_end] ; window end address (+1)
4740 <1> ; (addr of bottom right corner +1)
4741 0000D82C 779B <1> ja short sysvideo_15_14 ; out of display page
4742 <1> ; nothing to do
4743 0000D82E 89D3 <1> mov ebx, edx
4744 0000D830 C1EB10 <1> shr ebx, 16
4745 <1> ; ebx = row count
4746 0000D833 81E2FFFF0000 <1> and edx, 0FFFFh
4747 <1> ; edx = transfer count per row (from user's buffer)
4748 <1> ; (in pixels, window width)
4749 0000D839 89D0 <1> mov eax, edx
4750 0000D83B A3[8E9D0100] <1> mov [pixcount], eax ; 27/02/2021
4751 0000D840 E810000000 <1> call pixels_to_byte_count
4752 0000D845 89C1 <1> mov ecx, eax
4753 <1> ; ecx = transfer count per row (from user's buffer)
4754 <1> ; (in bytes, window width)
4755 0000D847 66A1[769D0100] <1> mov ax, [v_width]
4756 0000D84D E803000000 <1> call pixels_to_byte_count
4757 0000D852 89C2 <1> mov edx, eax
4758 <1> ; edx = byte count per row
4759 0000D854 C3 <1> retn ; cf = 0
4760 <1>
4761 <1> pixels_to_byte_count:
4762 <1> ; 29/01/2021
4763 <1> ; INPUT:
4764 <1> ; eax = pixel count
4765 <1> ; OUTPUT:
4766 <1> ; eax = byte count
4767 <1> ;
4768 0000D855 803D[799D0100]08 <1> cmp byte [v_bpp], 8
4769 0000D85C 7619 <1> jna short pixtobc_3 ; 8 bit colors
4770 0000D85E 803D[799D0100]18 <1> cmp byte [v_bpp], 24
4771 0000D865 720A <1> jb short pixtobc_1 ; 16 bit colors
4772 0000D867 770B <1> ja short pixtobc_2 ; 32 bit colors
4773 <1> ; 24 bit pixels
4774 <1> ; eax = eax * 3
4775 <1> ; push edx
4776 <1> ; mov edx, eax
4777 <1> ; shl eax, 1
4778 <1> ; add eax, edx
4779 <1> ; pop edx
4780 0000D869 50 <1> push eax
4781 0000D86A D1E0 <1> shl eax, 1
4782 0000D86C 010424 <1> add [esp], eax
4783 0000D86F 58 <1> pop eax
4784 0000D870 C3 <1> retn
4785 <1> pixtobc_1:
4786 <1> ; 32 bit pixels
4787 <1> ; eax = eax * 2
4788 0000D871 D1E0 <1> shl eax, 1
4789 0000D873 C3 <1> retn
4790 <1> pixtobc_2:
4791 <1> ; 16 bit pixels
4792 <1> ; eax = eax * 4
4793 0000D874 C1E002 <1> shl eax, 2
4794 <1> pixtobc_3:
4795 0000D877 C3 <1> retn
4796 <1>
4797 <1> calc_pixel_offset:
4798 <1> ; 29/01/2021
4799 <1> ; INPUT:

```

```

4800      <1>      ;   eax = pixel position (row, column)
4801      <1>      ;   OUTPUT:
4802      <1>      ;   eax = pixel offset (linear address)
4803      <1>      ;
4804      0000D878 52      <1>      push    edx
4805      0000D879 50      <1>      push    eax
4806      0000D87A C1E810  <1>      shr     eax, 16
4807      0000D87D 7409    <1>      jz      short cpixo_0
4808      <1>      ;   eax = row
4809      0000D87F 0FB715[769D0100] <1>      movzx   edx, word [v_width]
4810      0000D886 F7E2    <1>      mul     edx
4811      <1>      cpixo_0:
4812      <1>      ;   eax = row * screen width
4813      0000D888 5A      <1>      pop     edx
4814      0000D889 81E2FFFF0000 <1>      and     edx, 0FFFFh
4815      <1>      ;   edx = column
4816      0000D88F 01D0    <1>      add     eax, edx
4817      <1>      ;   eax = (row * screen width) + column
4818      0000D891 5A      <1>      pop     edx
4819      0000D892 C3      <1>      retn
4820      <1>
4821      <1>      ; 02/02/2021
4822      <1>      ; 29/01/2021
4823      <1>      pixel_ops:
4824      0000D893 [D3D80000] <1>      dd      pix_op_cpy ; copy pixels (user to system)
4825      0000D897 [1EDE0000] <1>      dd      pix_op_new ; change (new, fill) color
4826      0000D89B [3BD90000] <1>      dd      pix_op_add ; add color (up to 0FFh)
4827      0000D89F [EDD90000] <1>      dd      pix_op_sub ; sub color (down to 0)
4828      0000D8A3 [08DC0000] <1>      dd      pix_op_orc ; or color
4829      0000D8A7 [BADC0000] <1>      dd      pix_op_and ; and color
4830      0000D8AB [6CDD0000] <1>      dd      pix_op_xor ; xor color
4831      0000D8AF [E2DE0000] <1>      dd      pix_op_not ; not color
4832      0000D8B3 [8CDF0000] <1>      dd      pix_op_neg ; neg color
4833      0000D8B7 [36E00000] <1>      dd      pix_op_inc ; inc color
4834      0000D8BB [E0E00000] <1>      dd      pix_op_dec ; dec color
4835      0000D8BF [9FDA0000] <1>      dd      pix_op_mix ; mix color
4836      0000D8C3 [66DB0000] <1>      dd      pix_op_rpl ; replace color
4837      0000D8C7 [8AE10000] <1>      dd      pix_op_blk ; copy pixel block(s) (sys)
4838      0000D8CB [38E20000] <1>      dd      pix_op_lin ; write line(s)
4839      0000D8CF [1BE60000] <1>      dd      pix_op_chr ; write character (font)
4840      <1>
4841      <1>      pix_op_cpy:
4842      <1>      ; 21/02/2021
4843      <1>      ; 06/02/2021
4844      <1>      ; 30/01/2021
4845      <1>      ; COPY PIXELS
4846      <1>
4847      <1>      ; INPUT:
4848      <1>      ; If bit 4 of BL or [v_ops] = 1 -window copy-
4849      <1>      ; ECX = start position (row, column)
4850      <1>      ; (HW = row, CX = column)
4851      <1>      ; EDX = size (rows, columns)
4852      <1>      ; (HW = rows, DX = columns)
4853      <1>      ; (0 -> invalid
4854      <1>      ; (1 -> horizontal or vertical line)
4855      <1>      ; If bit 4 of BL or [v_ops] = 0 -full screen-
4856      <1>      ; ECX and EDX will not be used
4857      <1>      ; ESI = user's buffer address
4858      <1>      ; [maskcolor] = mask color (to be excluded)
4859      <1>
4860      <1>      ; OUTPUT:
4861      <1>      ; [u.r0] will be > 0 if succesful
4862      <1>
4863      0000D8D3 F605[789D0100]10 <1>      test    byte [v_ops], 10h ; display page or window ?
4864      0000D8DA 752E      <1>      jnz     short pix_op_cpy_w ; window
4865      <1>
4866      0000D8DC 8B3D[7A9D0100] <1>      mov     edi, [v_mem] ; 21/02/2021
4867      <1>
4868      <1>      ; Copy user's buffer content do display page
4869      <1>      ; (full screen copy)
4870      0000D8E2 A1[7E9D0100] <1>      mov     eax, [v_siz] ; video page size
4871      0000D8E7 E869FFFFFF <1>      call    pixels_to_byte_count
4872      0000D8EC 89C1      <1>      mov     ecx, eax ; transfer count
4873      <1>      ; esi = user's buffer address (virtual)
4874      0000D8EE F605[789D0100]20 <1>      test    byte [v_ops], 20h ; masked copy ?
4875      0000D8F5 7405      <1>      jz      short pix_op_cpy_0 ; no
4876      0000D8F7 E96F0F0000 <1>      jmp     m_pix_op_cpy ; copy pixels except mask color
4877      <1>      pix_op_cpy_0:
4878      <1>      ; esi = user buffer for full screen copy
4879      <1>      ; edi = start of video memory
4880      <1>      ; (start of display page)
4881      <1>      ; ecx = byte count (display page size in bytes)
4882      0000D8FC E854320000 <1>      call    transfer_from_user_buffer
4883      0000D901 7206      <1>      jc      short pix_op_cpy_1
4884      0000D903 890D[548E0100] <1>      mov     [u.r0], ecx
4885      <1>      pix_op_cpy_1:
4886      0000D909 C3      <1>      retn     ; 06/02/2021
4887      <1>
4888      <1>      pix_op_cpy_w:
4889      0000D90A E8BCFEFFFF <1>      call    sysvideo_15_12 ; window preparations
4890      0000D90F 72F8      <1>      jc      short pix_op_cpy_1
4891      <1>      ; ecx = bytes per row (to be applied)
4892      <1>      ; edx = screen width in bytes
4893      <1>      ; ebx = row count
4894      0000D911 8B3D[829D0100] <1>      mov     edi, [v_str]
4895      0000D917 F605[789D0100]20 <1>      test    byte [v_ops], 20h ; masked copy ?
4896      0000D91E 7405      <1>      jz      short pix_op_cpy_w_0 ; no
4897      0000D920 E909100000 <1>      jmp     m_pix_op_cpy_w ; window copy except mask color
4898      <1>      pix_op_cpy_w_0:
4899      <1>      ; esi = current row (virtual) addr in user's buff
4900      <1>      ; edi = window's current row address (video mem)
4901      <1>      ; ecx = transfer count per row
4902      0000D925 E82B320000 <1>      call    transfer_from_user_buffer
4903      0000D92A 72DD      <1>      jc      short pix_op_cpy_1
4904      0000D92C 010D[548E0100] <1>      add     [u.r0], ecx
4905      0000D932 4B      <1>      dec     ebx
4906      0000D933 74D4      <1>      jz      short pix_op_cpy_1 ; ok.
4907      <1>      ; next row
4908      0000D935 01CE      <1>      add     esi, ecx ; next row in user's buffer
4909      0000D937 01D7      <1>      add     edi, edx ; next row of window (system)
4910      0000D939 EBEA      <1>      jmp     short pix_op_cpy_w_0
4911      <1>
4912      <1>      pix_op_add:
4913      <1>      ; 31/01/2021
4914      <1>      ; 30/01/2021
4915      <1>      ; ADD COLOR
4916      <1>
4917      <1>      ; INPUT:
4918      <1>      ; CL = color (8 bit, 256 colors)
4919      <1>      ; ECX = color (16 bit and true colors)
4920      <1>      ; EDX = start position (row, column)
4921      <1>      ; (HW = row, DX = column)
4922      <1>      ; ESI = size (rows, columns)
4923      <1>      ; (HW = rows, SI = columns)

```

```

4924      <1>      ;
4925      <1>      ; [maskcolor] = mask color (to be excluded)
4926      <1>      ;
4927      <1>      ; OUTPUT:
4928      <1>      ; [u.r0] will be > 0 if succesful
4929      <1>      ;
4930      0000D93B F605[789D0100]10 <1>      test    byte [v_ops], 10h ; display page or window ?
4931      0000D942 7555 <1>      jnz     short pix_op_add_w ; window
4932      <1>      ;
4933      0000D944 8B3D[7A9D0100] <1>      mov     edi, [v_mem]
4934      0000D94A 89FE <1>      mov     esi, edi
4935      <1>      ; ecx = color (CL, CX, ECX)
4936      0000D94C 89C8 <1>      mov     eax, ecx
4937      0000D94E 8B0D[7E9D0100] <1>      mov     ecx, [v_siz] ; display page pixel count
4938      <1>      ;
4939      0000D954 F605[789D0100]20 <1>      test    byte [v_ops], 20h ; masked color adding ?
4940      0000D95B 7405 <1>      jz      short pix_op_add_0 ; no
4941      0000D95D E9CB100000 <1>      jmp     m_pix_op_add ; add color except mask color
4942      <1>      pix_op_add_0:
4943      0000D962 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
4944      0000D969 7707 <1>      ja      short pix_op_add_1
4945      <1>      ;
4946      <1>      ; 256 colors (8bpp)
4947      0000D96B E84C0A0000 <1>      call    pix_op_add_8
4948      0000D970 EB1E <1>      jmp     short pix_op_add_4
4949      <1>      ;
4950      <1>      pix_op_add_1:
4951      0000D972 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
4952      0000D979 7710 <1>      ja      short pix_op_add_3 ; 32bpp
4953      0000D97B 7207 <1>      jb      short pix_op_add_2 ; 16bpp
4954      <1>      ;
4955      <1>      ; 24 bit true colors
4956      0000D97D E85A0A0000 <1>      call    pix_op_add_24
4957      0000D982 EB0C <1>      jmp     short pix_op_add_4
4958      <1>      ;
4959      <1>      ; 65536 colors (16bpp)
4960      <1>      pix_op_add_2:
4961      0000D984 E8410A0000 <1>      call    pix_op_add_16
4962      0000D989 EB05 <1>      jmp     short pix_op_add_4
4963      <1>      ;
4964      <1>      ; 32 bit true colors
4965      <1>      pix_op_add_3:
4966      0000D98B E86C0A0000 <1>      call    pix_op_add_32
4967      <1>      pix_op_add_4:
4968      0000D990 29F7 <1>      sub     edi, esi
4969      0000D992 893D[548E0100] <1>      mov     [u.r0], edi
4970      <1>      pix_op_add_5:
4971      0000D998 C3 <1>      retn
4972      <1>      ;
4973      <1>      pix_op_add_w:
4974      <1>      ; 31/01/2021
4975      0000D999 51 <1>      push    ecx ; * ; color
4976      0000D99A 89D1 <1>      mov     ecx, edx ; win start pos
4977      0000D99C 89F2 <1>      mov     edx, esi ; size (rows, cols)
4978      0000D99E E828FEFFFF <1>      call    sysvideo_15_12 ; window preparations
4979      0000D9A3 58 <1>      pop     eax ; * ; color
4980      0000D9A4 72F2 <1>      jc      short pix_op_add_5
4981      <1>      ;
4982      0000D9A6 F605[789D0100]20 <1>      test    byte [v_ops], 20h ; masked color adding ?
4983      0000D9AD 7405 <1>      jz      short pix_op_add_w_0 ; no
4984      0000D9AF E927110000 <1>      jmp     m_pix_op_add_w
4985      <1>      ; window add color except mask color
4986      <1>      pix_op_add_w_0:
4987      <1>      ; ecx = bytes per row (to be applied)
4988      <1>      ; edx = screen width in bytes
4989      <1>      ; ebx = row count
4990      <1>      ; eax = color
4991      <1>      ;
4992      0000D9B4 8B3D[829D0100] <1>      mov     edi, [v_str]
4993      0000D9BA 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
4994      0000D9C1 7707 <1>      ja      short pix_op_add_w_1
4995      <1>      ;
4996      <1>      ; 256 colors (8bpp)
4997      0000D9C3 BD[BCE30000] <1>      mov     ebp, pix_op_add_8
4998      0000D9C8 EB1E <1>      jmp     short pix_op_add_w_4
4999      <1>      ;
5000      <1>      pix_op_add_w_1:
5001      0000D9CA 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
5002      0000D9D1 7710 <1>      ja      short pix_op_add_w_3 ; 32bpp
5003      0000D9D3 7207 <1>      jb      short pix_op_add_w_2 ; 16bpp
5004      <1>      ;
5005      <1>      ; 24 bit true colors
5006      0000D9D5 BD[DCE30000] <1>      mov     ebp, pix_op_add_24
5007      0000D9DA EB0C <1>      jmp     short pix_op_add_w_4
5008      <1>      ;
5009      <1>      ; 65536 colors (16bpp)
5010      <1>      pix_op_add_w_2:
5011      0000D9DC BD[CAE30000] <1>      mov     ebp, pix_op_add_16
5012      0000D9E1 EB05 <1>      jmp     short pix_op_add_w_4
5013      <1>      ;
5014      <1>      ; 32 bit true colors
5015      <1>      pix_op_add_w_3:
5016      0000D9E3 BD[FCE30000] <1>      mov     ebp, pix_op_add_32
5017      <1>      pix_op_add_w_4:
5018      0000D9E8 E95F010000 <1>      jmp     pix_op_add_w_x
5019      <1>      ;
5020      <1>      pix_op_sub:
5021      <1>      ; 31/01/2021
5022      <1>      ; SUB COLOR
5023      <1>      ;
5024      <1>      ; INPUT:
5025      <1>      ; CL = color (8 bit, 256 colors)
5026      <1>      ; ECX = color (16 bit and true colors)
5027      <1>      ; EDX = start position (row, column)
5028      <1>      ; (HW = row, DX = column)
5029      <1>      ; ESI = size (rows, columns)
5030      <1>      ; (HW = rows, SI = columns)
5031      <1>      ;
5032      <1>      ; [maskcolor] = mask color (to be excluded)
5033      <1>      ;
5034      <1>      ; OUTPUT:
5035      <1>      ; [u.r0] will be > 0 if succesful
5036      <1>      ;
5037      0000D9ED F605[789D0100]10 <1>      test    byte [v_ops], 10h ; display page or window ?
5038      0000D9F4 7555 <1>      jnz     short pix_op_sub_w ; window
5039      <1>      ;
5040      0000D9F6 8B3D[7A9D0100] <1>      mov     edi, [v_mem]
5041      0000D9FC 89FE <1>      mov     esi, edi
5042      <1>      ; ecx = color (CL, CX, ECX)
5043      0000D9FE 89C8 <1>      mov     eax, ecx
5044      0000DA00 8B0D[7E9D0100] <1>      mov     ecx, [v_siz] ; display page pixel count
5045      <1>      ;
5046      0000DA06 F605[789D0100]20 <1>      test    byte [v_ops], 20h ; masked color subtract ?
5047      0000DA0D 7405 <1>      jz      short pix_op_sub_0 ; no

```



```

5048 0000DA0F E9FA100000 <1> jmp m_pix_op_sub ; sub color except mask color
5049 <1> pix_op_sub_0:
5050 0000DA14 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
5051 0000DA1B 7707 <1> ja short pix_op_sub_1
5052 <1>
5053 <1> ; 256 colors (8bpp)
5054 0000DA1D E8E9090000 <1> call pix_op_sub_8
5055 0000DA22 EB1E <1> jmp short pix_op_sub_4
5056 <1>
5057 <1> pix_op_sub_1:
5058 0000DA24 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
5059 0000DA2B 7710 <1> ja short pix_op_sub_3 ; 32bpp
5060 0000DA2D 7207 <1> jb short pix_op_sub_2 ; 16bpp
5061 <1>
5062 <1> ; 24 bit true colors
5063 0000DA2F E8FA090000 <1> call pix_op_sub_24
5064 0000DA34 EB0C <1> jmp short pix_op_sub_4
5065 <1>
5066 <1> ; 65536 colors (16bpp)
5067 <1> pix_op_sub_2:
5068 0000DA36 E8E0090000 <1> call pix_op_sub_16
5069 0000DA3B EB05 <1> jmp short pix_op_sub_4
5070 <1>
5071 <1> ; 32 bit true colors
5072 <1> pix_op_sub_3:
5073 0000DA3D E8060A0000 <1> call pix_op_sub_32
5074 <1> pix_op_sub_4:
5075 0000DA42 29F7 <1> sub edi, esi
5076 0000DA44 893D[548E0100] <1> mov [u.r0], edi
5077 <1> pix_op_sub_5:
5078 0000DA4A C3 <1> retn
5079 <1>
5080 <1> pix_op_sub_w:
5081 <1> ; 31/01/2021
5082 0000DA4B 51 <1> push ecx ; * ; color
5083 0000DA4C 89D1 <1> mov ecx, edx ; win start pos
5084 0000DA4E 89F2 <1> mov edx, esi ; size (rows, cols)
5085 0000DA50 E876FDFFFF <1> call sysvideo_15_12 ; window preparations
5086 0000DA55 58 <1> pop eax ; * ; color
5087 0000DA56 72F2 <1> jc short pix_op_sub_5
5088 <1>
5089 0000DA58 F605[789D0100]20 <1> test byte [v_ops], 20h ; masked color subtract ?
5090 0000DA5F 7405 <1> jz short pix_op_sub_w_0 ; no
5091 0000DA61 E94B110000 <1> jmp m_pix_op_sub_w
5092 <1> ; window sub color except mask color
5093 <1> pix_op_sub_w_0:
5094 <1> ; ecx = bytes per row (to be applied)
5095 <1> ; edx = screen width in bytes
5096 <1> ; ebx = row count
5097 <1> ; eax = color
5098 <1>
5099 0000DA66 8B3D[829D0100] <1> mov edi, [v_str]
5100 0000DA6C 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
5101 0000DA73 7707 <1> ja short pix_op_sub_w_1
5102 <1>
5103 <1> ; 256 colors (8bpp)
5104 0000DA75 BD[0BE40000] <1> mov ebp, pix_op_sub_8
5105 0000DA7A EB1E <1> jmp short pix_op_sub_w_4
5106 <1>
5107 <1> pix_op_sub_w_1:
5108 0000DA7C 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
5109 0000DA83 7710 <1> ja short pix_op_sub_w_3 ; 32bpp
5110 0000DA85 7207 <1> jb short pix_op_sub_w_2 ; 16bpp
5111 <1>
5112 <1> ; 24 bit true colors
5113 0000DA87 BD[2EE40000] <1> mov ebp, pix_op_sub_24
5114 0000DA8C EB0C <1> jmp short pix_op_sub_w_4
5115 <1>
5116 <1> ; 65536 colors (16bpp)
5117 <1> pix_op_sub_w_2:
5118 0000DA8E BD[1BE40000] <1> mov ebp, pix_op_sub_16
5119 0000DA93 EB05 <1> jmp short pix_op_sub_w_4
5120 <1>
5121 <1> ; 32 bit true colors
5122 <1> pix_op_sub_w_3:
5123 0000DA95 BD[48E40000] <1> mov ebp, pix_op_sub_32
5124 <1> pix_op_sub_w_4:
5125 0000DA9A E9AD000000 <1> jmp pix_op_sub_w_x
5126 <1>
5127 <1> pix_op_mix:
5128 <1> ; 31/01/2021
5129 <1> ; MIX COLOR
5130 <1>
5131 <1> ; INPUT:
5132 <1> ; CL = color (8 bit, 256 colors)
5133 <1> ; ECX = color (16 bit and true colors)
5134 <1> ; EDX = start position (row, column)
5135 <1> ; (HW = row, DX = column)
5136 <1> ; ESI = size (rows, columns)
5137 <1> ; (HW = rows, SI = columns)
5138 <1>
5139 <1> ; [maskcolor] = mask color (to be excluded)
5140 <1>
5141 <1> ; OUTPUT:
5142 <1> ; [u.r0] will be > 0 if succesful
5143 <1>
5144 0000DA9F F605[789D0100]10 <1> test byte [v_ops], 10h ; display page or window ?
5145 0000DAA6 7555 <1> jnz short pix_op_mix_w ; window
5146 <1>
5147 0000DAA8 8B3D[7A9D0100] <1> mov edi, [v_mem]
5148 0000DAAE 89FE <1> mov esi, edi
5149 <1> ; ecx = color (CL, CX, ECX)
5150 0000DAB0 89C8 <1> mov eax, ecx
5151 0000DAB2 8B0D[7E9D0100] <1> mov ecx, [v_siz] ; display page pixel count
5152 <1>
5153 0000DAB8 F605[789D0100]20 <1> test byte [v_ops], 20h ; masked color mix ?
5154 0000DABF 7405 <1> jz short pix_op_mix_0 ; no
5155 0000DAC1 E91E110000 <1> jmp m_pix_op_mix ; mix colors except mask color
5156 <1> pix_op_mix_0:
5157 0000DAC6 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
5158 0000DACD 7707 <1> ja short pix_op_mix_1
5159 <1>
5160 <1> ; 256 colors (8bpp)
5161 0000DACF E8F3090000 <1> call pix_op_mix_8
5162 0000DAD4 EB1E <1> jmp short pix_op_mix_4
5163 <1>
5164 <1> pix_op_mix_1:
5165 0000DAD6 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
5166 0000DADD 7710 <1> ja short pix_op_mix_3 ; 32bpp
5167 0000DADE 7207 <1> jb short pix_op_mix_2 ; 16bpp
5168 <1>
5169 <1> ; 24 bit true colors
5170 0000DAE1 E8FC090000 <1> call pix_op_mix_24
5171 0000DAE6 EB0C <1> jmp short pix_op_mix_4

```

```

5172 <1>
5173 <1> ; 65536 colors (16bpp)
5174 <1> pix_op_mix_2:
5175 0000DAE8 E8E6090000 <1> call pix_op_mix_16
5176 0000DAED EB05 <1> jmp short pix_op_mix_4
5177 <1>
5178 <1> ; 32 bit true colors
5179 <1> pix_op_mix_3:
5180 0000DAEF E80A0A0000 <1> call pix_op_mix_32
5181 <1> pix_op_mix_4:
5182 0000DAF4 29F7 <1> sub edi, esi
5183 0000DAF6 893D[548E0100] <1> mov [u.r0], edi
5184 <1> pix_op_mix_5:
5185 0000DAFC C3 <1> retn
5186 <1>
5187 <1> pix_op_mix_w:
5188 <1> ; 31/01/2021
5189 0000DAFD 51 <1> push ecx ; * ; color
5190 0000DAFE 89D1 <1> mov ecx, edx ; win start pos
5191 0000DB00 89F2 <1> mov edx, esi ; size (rows, cols)
5192 0000DB02 E8C4FCFFFF <1> call sysvideo_15_12 ; window preparations
5193 0000DB07 58 <1> pop eax ; * ; color
5194 0000DB08 72F2 <1> jc short pix_op_mix_5
5195 <1>
5196 0000DB0A F605[789D0100]20 <1> test byte [v_ops], 20h ; masked color mix ?
5197 0000DB11 7405 <1> jz short pix_op_mix_w_0 ; no
5198 0000DB13 E969110000 <1> jmp m_pix_op_mix_w
5199 <1> ; window mix colors except mask color
5200 <1> pix_op_mix_w_0:
5201 <1> ; ecx = bytes per row (to be applied)
5202 <1> ; edx = screen width in bytes
5203 <1> ; ebx = row count
5204 <1> ; eax = color
5205 <1>
5206 0000DB18 8B3D[829D0100] <1> mov edi, [v_str]
5207 0000DB1E 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
5208 0000DB25 7707 <1> ja short pix_op_mix_w_1
5209 <1>
5210 <1> ; 256 colors (8bpp)
5211 0000DB27 BD[C7E40000] <1> mov ebp, pix_op_mix_8
5212 0000DB2C EB1E <1> jmp short pix_op_mix_w_x
5213 <1>
5214 <1> pix_op_mix_w_1:
5215 0000DB2E 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
5216 0000DB35 7710 <1> ja short pix_op_mix_w_3 ; 32bpp
5217 0000DB37 7207 <1> jb short pix_op_mix_w_2 ; 16bpp
5218 <1>
5219 <1> ; 24 bit true colors
5220 0000DB39 BD[E2E40000] <1> mov ebp, pix_op_mix_24
5221 0000DB3E EB0C <1> jmp short pix_op_mix_w_x
5222 <1>
5223 <1> ; 65536 colors (16bpp)
5224 <1> pix_op_mix_w_2:
5225 0000DB40 BD[D3E40000] <1> mov ebp, pix_op_mix_16
5226 0000DB45 EB05 <1> jmp short pix_op_mix_w_x
5227 <1>
5228 <1> ; 32 bit true colors
5229 <1> pix_op_mix_w_3:
5230 0000DB47 BD[FEE40000] <1> mov ebp, pix_op_mix_32
5231 <1> ; jmp short pix_op_mix_w_x
5232 <1>
5233 <1> pix_op_mix_w_x:
5234 <1> pix_op_add_w_x:
5235 <1> pix_op_sub_w_x:
5236 <1> pix_op_rpl_w_x:
5237 <1> pix_op_orc_w_x:
5238 <1> pix_op_and_w_x:
5239 <1> pix_op_xor_w_x:
5240 <1> ; 27/02/2021
5241 <1> ; 31/01/2021
5242 <1> ; ecx = bytes per row (to be applied)
5243 <1> ; edx = windows (screen) width in bytes
5244 <1> ; ebx = row count
5245 <1> ; eax = color
5246 <1> ; ebp = pixel operation subroutine address
5247 0000DB4C 52 <1> push edx
5248 0000DB4D 51 <1> push ecx
5249 0000DB4E 57 <1> push edi
5250 0000DB4F 8B0D[8E9D0100] <1> mov ecx, [pixcount] ; 27/02/2021
5251 0000DB55 FFD5 <1> call ebp ; call pixel-row operation
5252 0000DB57 5F <1> pop edi
5253 0000DB58 59 <1> pop ecx ; bytes per row
5254 0000DB59 010D[548E0100] <1> add [u.r0], ecx
5255 0000DB5F 5A <1> pop edx
5256 0000DB60 01D7 <1> add edi, edx ; next row
5257 0000DB62 4B <1> dec ebx
5258 0000DB63 75E7 <1> jnz short pix_op_mix_w_x
5259 0000DB65 C3 <1> retn
5260 <1>
5261 <1> pix_op_rpl:
5262 <1> ; 01/02/2021
5263 <1> ; REPLACE COLOR
5264 <1>
5265 <1> ; INPUT:
5266 <1> ; CL = old/current color (8 bit, 256 colors)
5267 <1> ; ECX = old/current color (16 bit and true colors)
5268 <1> ; DL = new color (8 bit, 256 colors)
5269 <1> ; EDX = new color (16 bit and true colors)
5270 <1> ; ESI = start position (row, column)
5271 <1> ; (HW = row, DX = column)
5272 <1> ; EDI = size (rows, columns)
5273 <1> ; (HW = rows, SI = columns)
5274 <1> ; OUTPUT:
5275 <1> ; [u.r0] will be > 0 if succesful
5276 <1>
5277 0000DB66 F605[789D0100]10 <1> test byte [v_ops], 10h ; display page or window ?
5278 0000DB6D 754D <1> jnz short pix_op_rpl_w ; window
5279 <1>
5280 0000DB6F 8B3D[7A9D0100] <1> mov edi, [v_mem]
5281 0000DB75 89FE <1> mov esi, edi
5282 <1> ; ecx = old color (CL, CX, ECX) -to be replaced with-
5283 <1> ; edx = new color (CL, CX, ECX) -new one-
5284 0000DB77 89D0 <1> mov eax, edx ; new color
5285 0000DB79 890D[8A9D0100] <1> mov [maskcolor], ecx ; old color
5286 0000DB7F 8B0D[7E9D0100] <1> mov ecx, [v_siz] ; display page pixel count
5287 <1> pix_op_rpl_0:
5288 0000DB85 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
5289 0000DB8C 7707 <1> ja short pix_op_rpl_1
5290 <1>
5291 <1> ; 256 colors (8bpp)
5292 0000DB8E E82F0A0000 <1> call pix_op_rpl_8
5293 0000DB93 EB1E <1> jmp short pix_op_rpl_4
5294 <1>
5295 <1> pix_op_rpl_1:

```

```

5296 0000DB95 803D[799D0100]18 <1>    cmp     byte [v_bpp], 24 ; 24bpp
5297 0000DB9C 7710 <1>    ja      short pix_op_rpl_3 ; 32bpp
5298 0000DB9E 7207 <1>    jb      short pix_op_rpl_2 ; 16bpp
5299 <1>
5300 <1>    ; 24 bit true colors
5301 0000DBA0 E840A0000 <1>    call    pix_op_rpl_24
5302 0000DBA5 EB0C <1>    jmp     short pix_op_rpl_4
5303 <1>
5304 <1>    ; 65536 colors (16bpp)
5305 <1> pix_op_rpl_2:
5306 0000DBA7 E8260A0000 <1>    call    pix_op_rpl_16
5307 0000DBAC EB05 <1>    jmp     short pix_op_rpl_4
5308 <1>
5309 <1>    ; 32 bit true colors
5310 <1> pix_op_rpl_3:
5311 0000DBAE E8540A0000 <1>    call    pix_op_rpl_32
5312 <1> pix_op_rpl_4:
5313 0000DBB3 29F7 <1>    sub     edi, esi
5314 0000DBB5 893D[548E0100] <1>    mov     [u.r0], edi
5315 <1> pix_op_rpl_5:
5316 0000DBBB C3 <1>    retn
5317 <1>
5318 <1> pix_op_rpl_w:
5319 <1>    ; 01/02/2021
5320 0000DBBC 890D[8A9D0100] <1>    mov     [maskcolor], ecx ; old color
5321 0000DBC2 52 <1>    push    edx ; * ; new color
5322 0000DBC3 89F1 <1>    mov     ecx, esi ; win start pos
5323 0000DBC5 89FA <1>    mov     edx, edi ; size (rows, cols)
5324 0000DBC7 E8FFFBFFFF <1>    call    sysvideo_15_12 ; window preparations
5325 0000DBCC 58 <1>    pop     eax ; * ; new color
5326 0000DBCD 72EC <1>    jc      short pix_op_rpl_5
5327 <1>
5328 <1>    ; replace window color
5329 <1> pix_op_rpl_w_0:
5330 <1>    ; ecx = bytes per row (to be applied)
5331 <1>    ; edx = screen width in bytes
5332 <1>    ; ebx = row count
5333 <1>    ; eax = new color
5334 <1>    ; [maskcolor] = old color
5335 <1>
5336 0000DBCF 8B3D[829D0100] <1>    mov     edi, [v_str]
5337 <1>
5338 0000DBD5 803D[799D0100]08 <1>    cmp     byte [v_bpp], 8 ; 8bpp
5339 0000DBDC 7707 <1>    ja      short pix_op_rpl_w_1
5340 <1>
5341 <1>    ; 256 colors (8bpp)
5342 0000DBDE BD[C2E50000] <1>    mov     ebp, pix_op_rpl_8
5343 0000DBE3 EB1E <1>    jmp     short pix_op_rpl_w_4
5344 <1>
5345 <1> pix_op_rpl_w_1:
5346 0000DBE5 803D[799D0100]18 <1>    cmp     byte [v_bpp], 24 ; 24bpp
5347 0000DBEC 7710 <1>    ja      short pix_op_rpl_w_3 ; 32bpp
5348 0000DBEE 7207 <1>    jb      short pix_op_rpl_w_2 ; 16bpp
5349 <1>
5350 <1>    ; 24 bit true colors
5351 0000DBF0 BD[E5E50000] <1>    mov     ebp, pix_op_rpl_24
5352 0000DBF5 EB0C <1>    jmp     short pix_op_rpl_w_4
5353 <1>
5354 <1>    ; 65536 colors (16bpp)
5355 <1> pix_op_rpl_w_2:
5356 0000DBF7 BD[D2E50000] <1>    mov     ebp, pix_op_rpl_16
5357 0000DBFC EB05 <1>    jmp     short pix_op_rpl_w_4
5358 <1>
5359 <1>    ; 32 bit true colors
5360 <1> pix_op_rpl_w_3:
5361 0000DBFE BD[07E60000] <1>    mov     ebp, pix_op_rpl_32
5362 <1> pix_op_rpl_w_4:
5363 0000DC03 E944FFFFFF <1>    jmp     pix_op_rpl_w_x
5364 <1>
5365 <1> pix_op_orc:
5366 <1>    ; 31/01/2021
5367 <1>    ; OR COLOR
5368 <1>
5369 <1>    ; INPUT:
5370 <1>    ; CL = color (8 bit, 256 colors)
5371 <1>    ; ECX = color (16 bit and true colors)
5372 <1>    ; EDX = start position (row, column)
5373 <1>    ; (HW = row, DX = column)
5374 <1>    ; ESI = size (rows, columns)
5375 <1>    ; (HW = rows, SI = columns)
5376 <1>
5377 <1>    ; [maskcolor] = mask color (to be excluded)
5378 <1>
5379 <1>    ; OUTPUT:
5380 <1>    ; [u.r0] will be > 0 if succesful
5381 <1>
5382 0000DC08 F605[789D0100]10 <1>    test    byte [v_ops], 10h ; display page or window ?
5383 0000DC0F 7555 <1>    jnz     short pix_op_or_w ; window
5384 <1>
5385 0000DC11 8B3D[7A9D0100] <1>    mov     edi, [v_mem]
5386 0000DC17 89FE <1>    mov     esi, edi
5387 <1>    ; ecx = color (CL, CX, ECX)
5388 0000DC19 89C8 <1>    mov     eax, ecx
5389 0000DC1B 8B0D[7E9D0100] <1>    mov     ecx, [v_siz] ; display page pixel count
5390 <1>
5391 0000DC21 F605[789D0100]20 <1>    test    byte [v_ops], 20h ; masked color 'or' ?
5392 0000DC28 7405 <1>    jz      short pix_op_or_0 ; no
5393 0000DC2A E945110000 <1>    jmp     m_pix_op_or ; 'or' color except mask color
5394 <1> pix_op_or_0:
5395 0000DC2F 803D[799D0100]08 <1>    cmp     byte [v_bpp], 8 ; 8bpp
5396 0000DC36 7707 <1>    ja      short pix_op_or_1
5397 <1>
5398 <1>    ; 256 colors (8bpp)
5399 0000DC38 E81B080000 <1>    call    pix_op_or_8
5400 0000DC3D EB1E <1>    jmp     short pix_op_or_4
5401 <1>
5402 <1> pix_op_or_1:
5403 0000DC3F 803D[799D0100]18 <1>    cmp     byte [v_bpp], 24 ; 24bpp
5404 0000DC46 7710 <1>    ja      short pix_op_or_3 ; 32bpp
5405 0000DC48 7207 <1>    jb      short pix_op_or_2 ; 16bpp
5406 <1>
5407 <1>    ; 24 bit true colors
5408 0000DC4A E817080000 <1>    call    pix_op_or_24
5409 0000DC4F EB0C <1>    jmp     short pix_op_or_4
5410 <1>
5411 <1>    ; 65536 colors (16bpp)
5412 <1> pix_op_or_2:
5413 0000DC51 E808080000 <1>    call    pix_op_or_16
5414 0000DC56 EB05 <1>    jmp     short pix_op_or_4
5415 <1>
5416 <1>    ; 32 bit true colors
5417 <1> pix_op_or_3:
5418 0000DC58 E818080000 <1>    call    pix_op_or_32
5419 <1> pix_op_or_4:

```

```

5420 0000DC5D 29F7      <1>      sub     edi, esi
5421 0000DC5F 893D[548E0100]  <1>      mov     [u.r0], edi
5422                                <1> pix_op_or_5:
5423 0000DC65 C3          <1>      retn
5424                                <1>
5425                                <1> pix_op_or_w:
5426                                <1>      ; 31/01/2021
5427 0000DC66 51          <1>      push    ecx ; * ; color
5428 0000DC67 89D1      <1>      mov     ecx, edx ; win start pos
5429 0000DC69 89F2      <1>      mov     edx, esi ; size (rows, cols)
5430 0000DC6B E85BF8FFFF <1>      call    sysvideo_15_12 ; window preparations
5431 0000DC70 58          <1>      pop     eax ; * ; color
5432 0000DC71 72F2      <1>      jc      short pix_op_or_5
5433                                <1>
5434 0000DC73 F605[789D0100]20 <1>      test    byte [v_ops], 20h ; masked color 'or' ?
5435 0000DC7A 7405      <1>      jz      short pix_op_or_w_0 ; no
5436 0000DC7C E980110000 <1>      jmp     m_pix_op_or_w
5437                                <1>      ; window 'or' color except mask color
5438                                <1> pix_op_or_w_0:
5439                                <1>      ; ecx = bytes per row (to be applied)
5440                                <1>      ; edx = screen width in bytes
5441                                <1>      ; ebx = row count
5442                                <1>      ; eax = color
5443                                <1>
5444 0000DC81 8B3D[829D0100] <1>      mov     edi, [v_str]
5445 0000DC87 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
5446 0000DC8E 7707      <1>      ja      short pix_op_or_w_1
5447                                <1>
5448                                <1>      ; 256 colors (8bpp)
5449 0000DC90 BD[58E40000] <1>      mov     ebp, pix_op_or_8
5450 0000DC95 EB1E      <1>      jmp     short pix_op_or_w_4
5451                                <1>
5452                                <1> pix_op_or_w_1:
5453 0000DC97 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
5454 0000DC9E 7710      <1>      ja      short pix_op_or_w_3 ; 32bpp
5455 0000DCA0 7207      <1>      jb      short pix_op_or_w_2 ; 16bpp
5456                                <1>
5457                                <1>      ; 24 bit true colors
5458 0000DCA2 BD[66E40000] <1>      mov     ebp, pix_op_or_24
5459 0000DCA7 EB0C      <1>      jmp     short pix_op_or_w_4
5460                                <1>
5461                                <1>      ; 65536 colors (16bpp)
5462                                <1> pix_op_or_w_2:
5463 0000DCA9 BD[5EE40000] <1>      mov     ebp, pix_op_or_16
5464 0000DCAE EB05      <1>      jmp     short pix_op_or_w_4
5465                                <1>
5466                                <1>      ; 32 bit true colors
5467                                <1> pix_op_or_w_3:
5468 0000DCB0 BD[75E40000] <1>      mov     ebp, pix_op_or_32
5469                                <1> pix_op_or_w_4:
5470 0000DCB5 E992FEFFFF <1>      jmp     pix_op_orc_w_x
5471                                <1>
5472                                <1> pix_op_and:
5473                                <1>      ; 31/01/2021
5474                                <1>      ; AND COLOR
5475                                <1>      ;
5476                                <1>      ; INPUT:
5477                                <1>      ; CL = color (8 bit, 256 colors)
5478                                <1>      ; ECX = color (16 bit and true colors)
5479                                <1>      ; EDX = start position (row, column)
5480                                <1>      ;      (HW = row, DX = column)
5481                                <1>      ; ESI = size (rows, cols)
5482                                <1>      ;      (HW = rows, SI = columns)
5483                                <1>      ;
5484                                <1>      ; [maskcolor] = mask color (to be excluded)
5485                                <1>      ;
5486                                <1>      ; OUTPUT:
5487                                <1>      ; [u.r0] will be > 0 if succesful
5488                                <1>
5489 0000DCBA F605[789D0100]10 <1>      test    byte [v_ops], 10h ; display page or window ?
5490 0000DCC1 7555      <1>      jnz     short pix_op_and_w ; window
5491                                <1>
5492 0000DCC3 8B3D[7A9D0100] <1>      mov     edi, [v_mem]
5493 0000DCC9 89FE      <1>      mov     esi, edi
5494                                <1>      ; ecx = color (CL, CX, ECX)
5495 0000DCCB 89C8      <1>      mov     eax, ecx
5496 0000DCCD 8B0D[7E9D0100] <1>      mov     ecx, [v_siz] ; display page pixel count
5497                                <1>
5498 0000DCD3 F605[789D0100]20 <1>      test    byte [v_ops], 20h ; masked color 'and' ?
5499 0000DCDA 7405      <1>      jz      short pix_op_and_0 ; no
5500 0000DCDC E9D30F0000 <1>      jmp     m_pix_op_and ; 'and' color except mask color
5501                                <1> pix_op_and_0:
5502 0000DCE1 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
5503 0000DCE8 7707      <1>      ja      short pix_op_and_1
5504                                <1>
5505                                <1>      ; 256 colors (8bpp)
5506 0000DCEA E88E070000 <1>      call    pix_op_and_8
5507 0000DCEF EB1E      <1>      jmp     short pix_op_and_4
5508                                <1>
5509                                <1> pix_op_and_1:
5510 0000DCF1 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
5511 0000DCF8 7710      <1>      ja      short pix_op_and_3 ; 32bpp
5512 0000DCFA 7207      <1>      jb      short pix_op_and_2 ; 16bpp
5513                                <1>
5514                                <1>      ; 24 bit true colors
5515 0000DCFC E88A070000 <1>      call    pix_op_and_24
5516 0000DD01 EB0C      <1>      jmp     short pix_op_and_4
5517                                <1>
5518                                <1>      ; 65536 colors (16bpp)
5519                                <1> pix_op_and_2:
5520 0000DD03 E87B070000 <1>      call    pix_op_and_16
5521 0000DD08 EB05      <1>      jmp     short pix_op_and_4
5522                                <1>
5523                                <1>      ; 32 bit true colors
5524                                <1> pix_op_and_3:
5525 0000DD0A E88B070000 <1>      call    pix_op_and_32
5526                                <1> pix_op_and_4:
5527 0000DD0F 29F7      <1>      sub     edi, esi
5528 0000DD11 893D[548E0100] <1>      mov     [u.r0], edi
5529                                <1> pix_op_and_5:
5530 0000DD17 C3          <1>      retn
5531                                <1>
5532                                <1> pix_op_and_w:
5533                                <1>      ; 31/01/2021
5534 0000DD18 51          <1>      push    ecx ; * ; color
5535 0000DD19 89D1      <1>      mov     ecx, edx ; win start pos
5536 0000DD1B 89F2      <1>      mov     edx, esi ; size (rows, cols)
5537 0000DD1D E8A9FAFFFF <1>      call    sysvideo_15_12 ; window preparations
5538 0000DD22 58          <1>      pop     eax ; * ; color
5539 0000DD23 72F2      <1>      jc      short pix_op_and_5
5540                                <1>
5541 0000DD25 F605[789D0100]20 <1>      test    byte [v_ops], 20h ; masked color 'and' ?
5542 0000DD2C 7405      <1>      jz      short pix_op_and_w_0 ; no
5543 0000DD2E E90E100000 <1>      jmp     m_pix_op_and_w

```

```

5544                                     <1>           ; window 'and' color except mask color
5545 pix_op_and_w_0:                       <1>
5546                                     <1>           ; ecx = bytes per row (to be applied)
5547                                     <1>           ; edx = screen width in bytes
5548                                     <1>           ; ebx = row count
5549                                     <1>           ; eax = color
5550                                     <1>
5551 0000DD33 8B3D[829D0100]               <1>         mov     edi, [v_str]
5552 0000DD39 803D[799D0100]08            <1>         cmp     byte [v_bpp], 8 ; 8bpp
5553 0000DD40 7707                        <1>         ja      short pix_op_and_w_1
5554                                     <1>
5555                                     <1>           ; 256 colors (8bpp)
5556 0000DD42 BD[7DE40000]                 <1>         mov     ebp, pix_op_and_8
5557 0000DD47 EB1E                        <1>         jmp     short pix_op_and_w_4
5558                                     <1>
5559 pix_op_and_w_1:                       <1>
5560 0000DD49 803D[799D0100]18            <1>         cmp     byte [v_bpp], 24 ; 24bpp
5561 0000DD50 7710                        <1>         ja      short pix_op_and_w_3 ; 32bpp
5562 0000DD52 7207                        <1>         jb      short pix_op_and_w_2 ; 16bpp
5563                                     <1>
5564                                     <1>           ; 24 bit true colors
5565 0000DD54 BD[8BE40000]                 <1>         mov     ebp, pix_op_and_24
5566 0000DD59 EB0C                        <1>         jmp     short pix_op_and_w_4
5567                                     <1>
5568                                     <1>           ; 65536 colors (16bpp)
5569 pix_op_and_w_2:                       <1>
5570 0000DD5B BD[83E40000]                 <1>         mov     ebp, pix_op_and_16
5571 0000DD60 EB05                        <1>         jmp     short pix_op_and_w_4
5572                                     <1>
5573                                     <1>           ; 32 bit true colors
5574 pix_op_and_w_3:                       <1>
5575 0000DD62 BD[9AE40000]                 <1>         mov     ebp, pix_op_and_32
5576                                     <1> pix_op_and_w_4:
5577 0000DD67 E9E0FDFFFF                  <1>         jmp     pix_op_and_w_x
5578                                     <1>
5579 pix_op_xor:                           <1>
5580                                     <1>           ; 31/01/2021
5581                                     <1>           ; XOR COLOR
5582                                     <1>
5583                                     <1>           ; INPUT:
5584                                     <1>           ; CL = color (8 bit, 256 colors)
5585                                     <1>           ; ECX = color (16 bit and true colors)
5586                                     <1>           ; EDX = start position (row, column)
5587                                     <1>           ;         (HW = row, DX = column)
5588                                     <1>           ; ESI = size (rows, columns)
5589                                     <1>           ;         (HW = rows, SI = columns)
5590                                     <1>
5591                                     <1>           ; [maskcolor] = mask color (to be excluded)
5592                                     <1>
5593                                     <1>           ; OUTPUT:
5594                                     <1>           ; [u.r0] will be > 0 if succesful
5595                                     <1>
5596 0000DD6C F605[789D0100]10            <1>         test    byte [v_ops], 10h ; display page or window ?
5597 0000DD73 7555                        <1>         jnz     short pix_op_xor_w ; window
5598                                     <1>
5599 0000DD75 8B3D[7A9D0100]               <1>         mov     edi, [v_mem]
5600 0000DD7B 89FE                        <1>         mov     esi, edi
5601                                     <1>           ; ecx = color (CL, CX, ECX)
5602 0000DD7D 89C8                        <1>         mov     eax, ecx
5603 0000DD7F 8B0D[7E9D0100]               <1>         mov     ecx, [v_siz] ; display page pixel count
5604                                     <1>
5605 0000DD85 F605[789D0100]20            <1>         test    byte [v_ops], 20h ; masked color 'xor' ?
5606 0000DD8C 7405                        <1>         jz      short pix_op_xor_0 ; no
5607 0000DD8E E9A1100000                  <1>         jmp     m_pix_op_xor ; 'xor' color except mask color
5608                                     <1> pix_op_xor_0:
5609 0000DD93 803D[799D0100]08            <1>         cmp     byte [v_bpp], 8 ; 8bpp
5610 0000DD9A 7707                        <1>         ja      short pix_op_xor_1
5611                                     <1>
5612                                     <1>           ; 256 colors (8bpp)
5613 0000DD9C E801070000                  <1>         call    pix_op_xor_8
5614 0000DDA1 EB1E                        <1>         jmp     short pix_op_xor_4
5615                                     <1>
5616 pix_op_xor_1:                         <1>
5617 0000DDA3 803D[799D0100]18            <1>         cmp     byte [v_bpp], 24 ; 24bpp
5618 0000DDAA 7710                        <1>         ja      short pix_op_xor_3 ; 32bpp
5619 0000DDAC 7207                        <1>         jb      short pix_op_xor_2 ; 16bpp
5620                                     <1>
5621                                     <1>           ; 24 bit true colors
5622 0000DDAE E8FD060000                  <1>         call    pix_op_xor_24
5623 0000DDB3 EB0C                        <1>         jmp     short pix_op_xor_4
5624                                     <1>
5625                                     <1>           ; 65536 colors (16bpp)
5626 pix_op_xor_2:                         <1>
5627 0000DDB5 E8EE060000                  <1>         call    pix_op_xor_16
5628 0000DDBA EB05                        <1>         jmp     short pix_op_xor_4
5629                                     <1>
5630                                     <1>           ; 32 bit true colors
5631 pix_op_xor_3:                         <1>
5632 0000DDBC E8FE060000                  <1>         call    pix_op_xor_32
5633                                     <1> pix_op_xor_4:
5634 0000DDC1 29F7                        <1>         sub     edi, esi
5635 0000DDC3 893D[548E0100]               <1>         mov     [u.r0], edi
5636                                     <1> pix_op_xor_5:
5637 0000DDC9 C3                          <1>         retn
5638                                     <1>
5639 pix_op_xor_w:                         <1>
5640                                     <1>           ; 31/01/2021
5641 0000DDCA 51                          <1>         push    ecx ; * ; color
5642 0000DDCB 89D1                        <1>         mov     ecx, edx ; win start pos
5643 0000DDCD 89F2                        <1>         mov     edx, esi ; size (rows, cols)
5644 0000DDCF E8F7F9FFFF                  <1>         call    sysvideo_15_12 ; window preparations
5645 0000DDD4 58                          <1>         pop     eax ; * ; color
5646 0000DDD5 72F2                        <1>         jc      short pix_op_xor_5
5647                                     <1>
5648 0000DDD7 F605[789D0100]20            <1>         test    byte [v_ops], 20h ; masked color 'xor' ?
5649 0000DDDE 7405                        <1>         jz      short pix_op_xor_w_0 ; no
5650 0000DDE0 E9DC100000                  <1>         jmp     m_pix_op_xor_w ; window 'xor' color except mask color
5651                                     <1>
5652 pix_op_xor_w_0:                       <1>
5653                                     <1>           ; ecx = bytes per row (to be applied)
5654                                     <1>           ; edx = screen width in bytes
5655                                     <1>           ; ebx = row count
5656                                     <1>           ; eax = color
5657                                     <1>
5658 0000DDE5 8B3D[829D0100]               <1>         mov     edi, [v_str]
5659 0000DDEB 803D[799D0100]08            <1>         cmp     byte [v_bpp], 8 ; 8bpp
5660 0000DDF2 7707                        <1>         ja      short pix_op_xor_w_1
5661                                     <1>
5662                                     <1>           ; 256 colors (8bpp)
5663 0000DDF4 BD[A2E40000]                 <1>         mov     ebp, pix_op_xor_8
5664 0000DDF9 EB1E                        <1>         jmp     short pix_op_xor_w_4
5665                                     <1>
5666                                     <1> pix_op_xor_w_1:
5667 0000DDFB 803D[799D0100]18            <1>         cmp     byte [v_bpp], 24 ; 24bpp

```

```

5668 0000DE02 7710      <1>      ja      short pix_op_xor_w_3 ; 32bpp
5669 0000DE04 7207      <1>      jb      short pix_op_xor_w_2 ; 16bpp
5670                                <1>
5671                                <1>      ; 24 bit true colors
5672 0000DE06 BD[B0E40000] <1>      mov     ebp, pix_op_xor_24
5673 0000DE0B EB0C      <1>      jmp     short pix_op_xor_w_4
5674                                <1>
5675                                <1>      ; 65536 colors (16bpp)
5676                                <1> pix_op_xor_w_2:
5677 0000DE0D BD[A8E40000] <1>      mov     ebp, pix_op_xor_16
5678 0000DE12 EB05      <1>      jmp     short pix_op_xor_w_4
5679                                <1>
5680                                <1>      ; 32 bit true colors
5681                                <1> pix_op_xor_w_3:
5682 0000DE14 BD[BFE40000] <1>      mov     ebp, pix_op_xor_32
5683                                <1> pix_op_xor_w_4:
5684 0000DE19 E92EFDFFFF <1>      jmp     pix_op_xor_w_x
5685                                <1>
5686                                <1> pix_op_new:
5687                                <1>      ; 31/01/2021
5688                                <1>      ; 30/01/2021
5689                                <1>      ; CHANGE COLOR
5690                                <1>
5691                                <1>      ; INPUT:
5692                                <1>      ; CL = color (8 bit, 256 colors)
5693                                <1>      ; ECX = color (16 bit and true colors)
5694                                <1>      ; EDX = start position (row, column)
5695                                <1>      ;      (HW = row, DX = column)
5696                                <1>      ; ESI = size (rows, columns)
5697                                <1>      ;      (HW = rows, SI = columns)
5698                                <1>
5699                                <1>      ; [maskcolor] = mask color (to be excluded)
5700                                <1>
5701                                <1>      ; OUTPUT:
5702                                <1>      ; [u.r0] will be > 0 if succesful
5703                                <1>
5704 0000DE1E F605[789D0100]10 <1>      test    byte [v_ops], 10h ; display page or window ?
5705 0000DE25 7554      <1>      jnz     short pix_op_new_w ; window
5706                                <1>
5707 0000DE27 8B3D[7A9D0100] <1>      mov     edi, [v_mem]
5708 0000DE2D 89FE      <1>      mov     esi, edi
5709                                <1>      ; ecx = color (CL, CX, ECX)
5710 0000DE2F 89C8      <1>      mov     eax, ecx
5711 0000DE31 8B0D[7E9D0100] <1>      mov     ecx, [v_siz] ; display page pixel count
5712                                <1>
5713 0000DE37 F605[789D0100]20 <1>      test    byte [v_ops], 20h ; masked color change ?
5714 0000DE3E 7405      <1>      jz      short pix_op_new_0 ; no
5715 0000DE40 E90A0B0000 <1>      jmp     m_pix_op_new ; change color except mask color
5716                                <1> pix_op_new_0:
5717 0000DE45 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
5718 0000DE4C 7706      <1>      ja      short pix_op_new_2
5719                                <1>
5720                                <1>      ; 256 colors (8bpp)
5721                                <1> pix_op_new_1:
5722 0000DE4E 88C4      <1>      mov     ah, al
5723 0000DE50 D1E9      <1>      shr     ecx, 1
5724 0000DE52 EB12      <1>      jmp     short pix_op_new_3
5725                                <1>
5726                                <1> pix_op_new_2:
5727 0000DE54 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
5728 0000DE5B 7713      <1>      ja      short pix_op_new_4 ; 32bpp
5729 0000DE5D 7207      <1>      jb      short pix_op_new_3 ; 16bpp
5730                                <1>
5731                                <1>      ; 31/01/2021
5732                                <1>
5733                                <1>      ; 24 bit true colors
5734 0000DE5F E849050000 <1>      call    pix_op_new_24
5735                                <1>
5736 0000DE64 EB0C      <1>      jmp     short pix_op_new_5
5737                                <1>
5738                                <1>      ; 65536 colors (16bpp)
5739                                <1> pix_op_new_3:
5740 0000DE66 89C2      <1>      mov     edx, eax
5741 0000DE68 C1E010 <1>      shl     eax, 16
5742 0000DE6B 6689D0 <1>      mov     ax, dx
5743 0000DE6E D1E9      <1>      shr     ecx, 1 ; dword counts
5744                                <1>      ; 32 bit true colors
5745                                <1> pix_op_new_4:
5746 0000DE70 F3AB      <1>      rep     stosd
5747                                <1> pix_op_new_5:
5748 0000DE72 29F7      <1>      sub     edi, esi
5749 0000DE74 893D[548E0100] <1>      mov     [u.r0], edi
5750                                <1> pix_op_new_6:
5751 0000DE7A C3      <1>      retn
5752                                <1>
5753                                <1> pix_op_new_w:
5754                                <1>      ; 31/01/2021
5755                                <1>      ; 30/01/2021
5756 0000DE7B 51      <1>      push    ecx ; * ; color
5757 0000DE7C 89D1      <1>      mov     ecx, edx ; win start pos
5758 0000DE7E 89F2      <1>      mov     edx, esi ; size (rows, cols)
5759 0000DE80 E846F9FFFF <1>      call    sysvideo_15_12 ; window preparations
5760 0000DE85 58      <1>      pop     eax ; * ; color
5761 0000DE86 72F2      <1>      jc      short pix_op_new_6
5762                                <1>
5763 0000DE88 F605[789D0100]20 <1>      test    byte [v_ops], 20h ; masked color change ?
5764 0000DE8F 7405      <1>      jz      short pix_op_new_w_0 ; no
5765 0000DE91 E9470B0000 <1>      jmp     m_pix_op_new_w
5766                                <1>      ; window chg color except mask color
5767                                <1> pix_op_new_w_0:
5768                                <1>      ; ecx = bytes per row (to be applied)
5769                                <1>      ; edx = screen width in bytes
5770                                <1>      ; ebx = row count
5771                                <1>      ; eax = color
5772                                <1>
5773 0000DE96 8B3D[829D0100] <1>      mov     edi, [v_str]
5774                                <1>
5775 0000DE9C 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
5776 0000DEA3 7707      <1>      ja      short pix_op_new_w_1
5777                                <1>
5778                                <1>      ; 256 colors (8bpp)
5779 0000DEA5 BD[A6E30000] <1>      mov     ebp, pix_op_new_8
5780 0000DEAA EB1E      <1>      jmp     short pix_op_new_w_x
5781                                <1>
5782                                <1> pix_op_new_w_1:
5783 0000DEAC 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
5784 0000DEB3 7710      <1>      ja      short pix_op_new_w_3 ; 32bpp
5785 0000DEB5 7207      <1>      jb      short pix_op_new_w_2 ; 16bpp
5786                                <1>
5787                                <1>      ; 24 bit true colors
5788 0000DEB7 BD[ADE30000] <1>      mov     ebp, pix_op_new_24
5789 0000DEBC EB0C      <1>      jmp     short pix_op_new_w_x
5790                                <1>
5791                                <1>      ; 65536 colors (16bpp)

```

```

5792 <1> pix_op_new_w_2:
5793 0000DEBE BD[A9E30000] <1> mov ebp, pix_op_new_16
5794 0000DEC3 EB05 <1> jmp short pix_op_new_w_x
5795 <1>
5796 <1> ; 32 bit true colors
5797 <1> pix_op_new_w_3:
5798 0000DEC5 BD[B9E30000] <1> mov ebp, pix_op_new_32
5799 <1> ; jmp short pix_op_new_w_x
5800 <1>
5801 <1> pix_op_new_w_x:
5802 <1> pix_op_not_w_x:
5803 <1> pix_op_neg_w_x:
5804 <1> pix_op_inc_w_x:
5805 <1> pix_op_dec_w_x:
5806 <1> ; 27/02/2021
5807 <1> ; 01/02/2021
5808 <1> ; 31/01/2021
5809 <1> ; ecx = bytes per row (to be applied)
5810 <1> ; edx = windows (screen) width in bytes
5811 <1> ; ebx = row count
5812 <1> ; eax = color
5813 <1> ; ebp = pixel operation subroutine address
5814 <1> ; push edx ; 01/02/2021
5815 0000DECA 51 <1> push ecx
5816 0000DECB 57 <1> push edi
5817 0000DECC 8B0D[8E9D0100] <1> mov ecx, [pixcount] ; 27/02/2021
5818 0000DED2 FFD5 <1> call ebp ; call pixel-row operation
5819 0000DED4 5F <1> pop edi
5820 0000DED5 59 <1> pop ecx ; bytes per row
5821 0000DED6 010D[548E0100] <1> add [u.r0], ecx
5822 <1> ; pop edx ; 01/02/2021
5823 0000DEDC 01D7 <1> add edi, edx ; next row
5824 0000DEDE 4B <1> dec ebx
5825 0000DEDF 75E9 <1> jnz short pix_op_new_w_x
5826 0000DEE1 C3 <1> retn
5827 <1>
5828 <1> pix_op_not:
5829 <1> ; 31/01/2021
5830 <1> ; NOT COLOR
5831 <1>
5832 <1> ; INPUT:
5833 <1> ; ECX = start position (row, column)
5834 <1> ; (HW = row, CX = column)
5835 <1> ; EDX = size (rows, columns)
5836 <1> ; (HW = rows, DX = columns)
5837 <1> ; (0 -> invalid
5838 <1> ; (1 -> horizontal or vertical line)
5839 <1> ; [maskcolor] = mask color (to be excluded)
5840 <1>
5841 <1> ; OUTPUT:
5842 <1> ; [u.r0] will be > 0 if succesful
5843 <1>
5844 0000DEE2 F605[789D0100]10 <1> test byte [v_ops], 10h ; display page or window ?
5845 0000DEE9 7553 <1> jnz short pix_op_not_w ; window
5846 <1>
5847 0000DEEB 8B3D[7A9D0100] <1> mov edi, [v_mem]
5848 0000DEF1 89FE <1> mov esi, edi
5849 0000DEF3 8B0D[7E9D0100] <1> mov ecx, [v_siz] ; display page pixel count
5850 <1>
5851 0000DEF9 F605[789D0100]20 <1> test byte [v_ops], 20h ; masked color 'not' ?
5852 0000DF00 7405 <1> jz short pix_op_not_0 ; no
5853 0000DF02 E9ED0F0000 <1> jmp m_pix_op_not ; 'not' color except mask color
5854 <1> pix_op_not_0:
5855 0000DF07 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
5856 0000DF0E 7707 <1> ja short pix_op_not_1
5857 <1>
5858 <1> ; 256 colors (8bpp)
5859 0000DF10 E8F5050000 <1> call pix_op_not_8
5860 0000DF15 EB1E <1> jmp short pix_op_not_4
5861 <1>
5862 <1> pix_op_not_1:
5863 0000DF17 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
5864 0000DF1E 7710 <1> ja short pix_op_not_3 ; 32bpp
5865 0000DF20 7207 <1> jb short pix_op_not_2 ; 16bpp
5866 <1>
5867 <1> ; 24 bit true colors
5868 0000DF22 E8F1050000 <1> call pix_op_not_24
5869 0000DF27 EB0C <1> jmp short pix_op_not_4
5870 <1>
5871 <1> ; 65536 colors (16bpp)
5872 <1> pix_op_not_2:
5873 0000DF29 E8E2050000 <1> call pix_op_not_16
5874 0000DF2E EB05 <1> jmp short pix_op_not_4
5875 <1>
5876 <1> ; 32 bit true colors
5877 <1> pix_op_not_3:
5878 0000DF30 E8EE050000 <1> call pix_op_not_32
5879 <1> pix_op_not_4:
5880 0000DF35 29F7 <1> sub edi, esi
5881 0000DF37 893D[548E0100] <1> mov [u.r0], edi
5882 <1> pix_op_not_5:
5883 0000DF3D C3 <1> retn
5884 <1>
5885 <1> pix_op_not_w:
5886 <1> ; 31/01/2021
5887 <1> ; ecx = win start pos (row, column)
5888 <1> ; edx = size (rows, columns)
5889 0000DF3E E88F8FFFFF <1> call sysvideo_15_12 ; window preparations
5890 0000DF43 72F8 <1> jc short pix_op_not_5
5891 <1>
5892 0000DF45 F605[789D0100]20 <1> test byte [v_ops], 20h ; masked color 'not' ?
5893 0000DF4C 7405 <1> jz short pix_op_not_w_0 ; no
5894 0000DF4E E926100000 <1> jmp m_pix_op_not_w
5895 <1> ; window 'not' color except mask color
5896 <1> pix_op_not_w_0:
5897 <1> ; ecx = bytes per row (to be applied)
5898 <1> ; edx = screen width in bytes
5899 <1> ; ebx = row count
5900 <1>
5901 0000DF53 8B3D[829D0100] <1> mov edi, [v_str]
5902 <1>
5903 0000DF59 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
5904 0000DF60 7707 <1> ja short pix_op_not_w_1
5905 <1>
5906 <1> ; 256 colors (8bpp)
5907 0000DF62 BD[0AE50000] <1> mov ebp, pix_op_not_8
5908 0000DF67 EB1E <1> jmp short pix_op_not_w_4
5909 <1>
5910 <1> pix_op_not_w_1:
5911 0000DF69 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
5912 0000DF70 7710 <1> ja short pix_op_not_w_3 ; 32bpp
5913 0000DF72 7207 <1> jb short pix_op_not_w_2 ; 16bpp
5914 <1>
5915 <1> ; 24 bit true colors

```

```

5916 0000DF74 BD[18E50000] <1> mov ebp, pix_op_not_24
5917 0000DF79 EB0C <1> jmp short pix_op_not_w_4
5918 <1>
5919 <1> ; 65536 colors (16bpp)
5920 <1> pix_op_not_w_2:
5921 0000DF7B BD[10E50000] <1> mov ebp, pix_op_not_16
5922 0000DF80 EB05 <1> jmp short pix_op_not_w_4
5923 <1>
5924 <1> ; 32 bit true colors
5925 <1> pix_op_not_w_3:
5926 0000DF82 BD[23E50000] <1> mov ebp, pix_op_not_32
5927 <1> pix_op_not_w_4:
5928 0000DF87 E93EFFFFFF <1> jmp pix_op_not_w_x
5929 <1>
5930 <1> pix_op_neg:
5931 <1> ; 31/01/2021
5932 <1> ; NEGATE COLOR
5933 <1> ;
5934 <1> ; INPUT:
5935 <1> ; ECX = start position (row, column)
5936 <1> ; (HW = row, CX = column)
5937 <1> ; EDX = size (rows, columns)
5938 <1> ; (HW = rows, DX = columns)
5939 <1> ; (0 -> invalid
5940 <1> ; (1 -> horizontal or vertical line)
5941 <1> ; [maskcolor] = mask color (to be excluded)
5942 <1> ;
5943 <1> ; OUTPUT:
5944 <1> ; [u.r0] will be > 0 if succesful
5945 <1>
5946 0000DF8C F605[789D0100]10 <1> test byte [v_ops], 10h ; display page or window ?
5947 0000DF93 7553 <1> jnz short pix_op_neg_w ; window
5948 <1>
5949 0000DF95 8B3D[7A9D0100] <1> mov edi, [v_mem]
5950 0000DF9B 89FE <1> mov esi, edi
5951 0000DF9D 8B0D[7E9D0100] <1> mov ecx, [v_siz] ; display page pixel count
5952 <1>
5953 0000DFA3 F605[789D0100]20 <1> test byte [v_ops], 20h ; masked negate color ?
5954 0000DFAA 7405 <1> jz short pix_op_neg_0 ; no
5955 0000DFAC E9FB0F0000 <1> jmp m_pix_op_neg ; 'neg' color except mask color
5956 <1> pix_op_neg_0:
5957 0000DFB1 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
5958 0000DFB8 7707 <1> ja short pix_op_neg_1
5959 <1>
5960 <1> ; 256 colors (8bpp)
5961 0000DFBA E86C050000 <1> call pix_op_neg_8
5962 0000DFBF EB1E <1> jmp short pix_op_neg_4
5963 <1>
5964 <1> pix_op_neg_1:
5965 0000DFC1 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
5966 0000DFC8 7710 <1> ja short pix_op_neg_3 ; 32bpp
5967 0000DFCA 7207 <1> jb short pix_op_neg_2 ; 16bpp
5968 <1>
5969 <1> ; 24 bit true colors
5970 0000DFCC E868050000 <1> call pix_op_neg_24
5971 0000DFD1 EB0C <1> jmp short pix_op_neg_4
5972 <1>
5973 <1> ; 65536 colors (16bpp)
5974 <1> pix_op_neg_2:
5975 0000DFD3 E859050000 <1> call pix_op_neg_16
5976 0000DFD8 EB05 <1> jmp short pix_op_neg_4
5977 <1>
5978 <1> ; 32 bit true colors
5979 <1> pix_op_neg_3:
5980 0000DFDA E86C050000 <1> call pix_op_neg_32
5981 <1> pix_op_neg_4:
5982 0000DFDF 29F7 <1> sub edi, esi
5983 0000DFE1 893D[548E0100] <1> mov [u.r0], edi
5984 <1> pix_op_neg_5:
5985 0000DFE7 C3 <1> retn
5986 <1>
5987 <1> pix_op_neg_w:
5988 <1> ; 31/01/2021
5989 <1> ; ecx = win start pos (row, column)
5990 <1> ; edx = size (rows, columns)
5991 0000DFE8 E8DEF7FFFF <1> call sysvideo_15_12 ; window preparations
5992 0000DFED 72F8 <1> jc short pix_op_neg_5
5993 <1>
5994 0000DFEF F605[789D0100]20 <1> test byte [v_ops], 20h ; masked negate color ?
5995 0000DFFF 7405 <1> jz short pix_op_neg_w_0 ; no
5996 0000DF8F E934100000 <1> jmp m_pix_op_neg_w ; window 'neg' color except mask color
5997 <1>
5998 <1> pix_op_neg_w_0:
5999 <1> ; ecx = bytes per row (to be applied)
6000 <1> ; edx = screen width in bytes
6001 <1> ; ebx = row count
6002 <1>
6003 0000DFFD 8B3D[829D0100] <1> mov edi, [v_str]
6004 <1>
6005 0000E003 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
6006 0000E00A 7707 <1> ja short pix_op_neg_w_1
6007 <1>
6008 <1> ; 256 colors (8bpp)
6009 0000E00C BD[2BE50000] <1> mov ebp, pix_op_neg_8
6010 0000E011 EB1E <1> jmp short pix_op_neg_w_4
6011 <1>
6012 <1> pix_op_neg_w_1:
6013 0000E013 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
6014 0000E01A 7710 <1> ja short pix_op_neg_w_3 ; 32bpp
6015 0000E01C 7207 <1> jb short pix_op_neg_w_2 ; 16bpp
6016 <1>
6017 <1> ; 24 bit true colors
6018 0000E01E BD[39E50000] <1> mov ebp, pix_op_neg_24
6019 0000E023 EB0C <1> jmp short pix_op_neg_w_4
6020 <1>
6021 <1> ; 65536 colors (16bpp)
6022 <1> pix_op_neg_w_2:
6023 0000E025 BD[31E50000] <1> mov ebp, pix_op_neg_16
6024 0000E02A EB05 <1> jmp short pix_op_neg_w_4
6025 <1>
6026 <1> ; 32 bit true colors
6027 <1> pix_op_neg_w_3:
6028 0000E02C BD[4BE50000] <1> mov ebp, pix_op_neg_32
6029 <1> pix_op_neg_w_4:
6030 0000E031 E994FEFFFF <1> jmp pix_op_neg_w_x
6031 <1>
6032 <1> pix_op_inc:
6033 <1> ; 31/01/2021
6034 <1> ; INCREASE COLOR
6035 <1> ;
6036 <1> ; INPUT:
6037 <1> ; ECX = start position (row, column)
6038 <1> ; (HW = row, CX = column)
6039 <1> ; EDX = size (rows, columns)

```



```

6040      ;      (HW = rows, DX = columns)
6041      ;      (0 -> invalid
6042      ;      (1 -> horizontal or vertical line)
6043      ;      [maskcolor] = mask color (to be excluded)
6044      ;
6045      ; OUTPUT:
6046      ;      [u.r0] will be > 0 if succesful
6047
6048 0000E036 F605[789D0100]10      <1>      test    byte [v_ops], 10h ; display page or window ?
6049 0000E03D 7553                  <1>      jnz     short pix_op_inc_w ; window
6050
6051 0000E03F 8B3D[7A9D0100]      <1>      mov     edi, [v_mem]
6052 0000E045 89FE                  <1>      mov     esi, edi
6053 0000E047 8B0D[7E9D0100]      <1>      mov     ecx, [v_siz] ; display page pixel count
6054
6055 0000E04D F605[789D0100]20      <1>      test    byte [v_ops], 20h ; masked increase color ?
6056 0000E054 7405                  <1>      jz      short pix_op_inc_0 ; no
6057 0000E056 E909100000          <1>      jmp     m_pix_op_inc ; 'inc' color except mask color
6058
6059 0000E05B 803D[799D0100]08      <1> pix_op_inc_0:
6060 0000E062 7707                  <1>      cmp     byte [v_bpp], 8 ; 8bpp
6061                                     <1>      ja      short pix_op_inc_1
6062                                     <1>
6063                                     <1>      ; 256 colors (8bpp)
6064 0000E064 E8EA040000          <1>      call    pix_op_inc_8
6065 0000E069 EB1E                  <1>      jmp     short pix_op_inc_4
6066
6067 0000E06B 803D[799D0100]18      <1> pix_op_inc_1:
6068 0000E072 7710                  <1>      cmp     byte [v_bpp], 24 ; 24bpp
6069 0000E074 7207                  <1>      ja      short pix_op_inc_3 ; 32bpp
6070                                     <1>      jb      short pix_op_inc_2 ; 16bpp
6071                                     <1>
6072                                     <1>      ; 24 bit true colors
6073 0000E07B EB0C                  <1>      call    pix_op_inc_24
6074                                     <1>      jmp     short pix_op_inc_4
6075                                     <1>
6076                                     <1>      ; 65536 colors (16bpp)
6077 0000E07D E8DB040000          <1> pix_op_inc_2:
6078 0000E082 EB05                  <1>      call    pix_op_inc_16
6079                                     <1>      jmp     short pix_op_inc_4
6080                                     <1>
6081                                     <1>      ; 32 bit true colors
6082 0000E084 E8F5040000          <1> pix_op_inc_3:
6083                                     <1>      call    pix_op_inc_32
6084 0000E089 29F7                  <1> pix_op_inc_4:
6085 0000E08B 893D[548E0100]      <1>      sub     edi, esi
6086                                     <1>      mov     [u.r0], edi
6087 0000E091 C3                    <1> pix_op_inc_5:
6088                                     <1>      retn
6089
6090                                     <1> pix_op_inc_w:
6091                                     <1>      ; 31/01/2021
6092                                     <1>      ; ecx = win start pos (row, column)
6093 0000E092 E834F7FFFF          <1>      ; edx = size (rows, columns)
6094 0000E097 72F8                  <1>      call    sysvideo_15_12 ; window preparations
6095                                     <1>      jc      short pix_op_inc_5
6096
6097 0000E099 F605[789D0100]20      <1>      test    byte [v_ops], 20h ; masked increase color ?
6098 0000E0A0 7405                  <1>      jz      short pix_op_inc_w_0 ; no
6099 0000E0A2 E956100000          <1>      jmp     m_pix_op_inc_w
6100                                     <1>      ; window 'inc' color except mask color
6101                                     <1> pix_op_inc_w_0:
6102                                     <1>      ; ecx = bytes per row (to be applied)
6103                                     <1>      ; edx = screen width in bytes
6104                                     <1>      ; ebx = row count
6105 0000E0A7 8B3D[829D0100]      <1>      mov     edi, [v_str]
6106
6107 0000E0AD 803D[799D0100]08      <1>      cmp     byte [v_bpp], 8 ; 8bpp
6108 0000E0B4 7707                  <1>      ja      short pix_op_inc_w_1
6109                                     <1>
6110                                     <1>      ; 256 colors (8bpp)
6111 0000E0B6 BD[53E50000]          <1>      mov     ebp, pix_op_inc_8
6112 0000E0BB EB1E                  <1>      jmp     short pix_op_inc_w_4
6113
6114                                     <1> pix_op_inc_w_1:
6115 0000E0BD 803D[799D0100]18      <1>      cmp     byte [v_bpp], 24 ; 24bpp
6116 0000E0C4 7710                  <1>      ja      short pix_op_inc_w_3 ; 32bpp
6117 0000E0C6 7207                  <1>      jb      short pix_op_inc_w_2 ; 16bpp
6118                                     <1>
6119                                     <1>      ; 24 bit true colors
6120 0000E0C8 BD[6AE50000]          <1>      mov     ebp, pix_op_inc_24
6121 0000E0CD EB0C                  <1>      jmp     short pix_op_inc_w_4
6122                                     <1>
6123                                     <1>      ; 65536 colors (16bpp)
6124                                     <1> pix_op_inc_w_2:
6125 0000E0CF BD[5DE50000]          <1>      mov     ebp, pix_op_inc_16
6126 0000E0D4 EB05                  <1>      jmp     short pix_op_inc_w_4
6127                                     <1>
6128                                     <1>      ; 32 bit true colors
6129 0000E0D6 BD[7EE50000]          <1> pix_op_inc_w_3:
6130                                     <1>      mov     ebp, pix_op_inc_32
6131                                     <1> pix_op_inc_w_4:
6132 0000E0DB E9EAFDFFFF          <1>      jmp     pix_op_inc_w_x
6133
6134                                     <1> pix_op_dec:
6135                                     <1>      ; 31/01/2021
6136                                     <1>      ; DECREASE COLOR
6137                                     <1>
6138                                     <1>      ; INPUT:
6139                                     <1>      ; ECX = start position (row, column)
6140                                     <1>      ;      (HW = row, CX = column)
6141                                     <1>      ; EDX = size (rows, cols)
6142                                     <1>      ;      (HW = rows, DX = columns)
6143                                     <1>      ;      (0 -> invalid
6144                                     <1>      ;      (1 -> horizontal or vertical line)
6145                                     <1>      ;      [maskcolor] = mask color (to be excluded)
6146                                     <1>
6147                                     <1>      ; OUTPUT:
6148                                     <1>      ;      [u.r0] will be > 0 if succesful
6149
6150 0000E0E0 F605[789D0100]10      <1>      test    byte [v_ops], 10h ; display page or window ?
6151 0000E0E7 7553                  <1>      jnz     short pix_op_dec_w ; window
6152
6153 0000E0E9 8B3D[7A9D0100]      <1>      mov     edi, [v_mem]
6154 0000E0EF 89FE                  <1>      mov     esi, edi
6155 0000E0F1 8B0D[7E9D0100]      <1>      mov     ecx, [v_siz] ; display page pixel count
6156
6157 0000E0F7 F605[789D0100]20      <1>      test    byte [v_ops], 20h ; masked decrease color ?
6158 0000E0FE 7405                  <1>      jz      short pix_op_dec_0 ; no
6159 0000E100 E92B100000          <1>      jmp     m_pix_op_dec ; 'dec' color except mask color
6160
6161 0000E105 803D[799D0100]08      <1> pix_op_dec_0:
6162 0000E10C 7707                  <1>      cmp     byte [v_bpp], 8 ; 8bpp
6163                                     <1>      ja      short pix_op_dec_1

```

```

6164      ; 256 colors (8bpp)
6165 0000E10E E877040000    <1>    call    pix_op_dec_8
6166 0000E113 EB1E        <1>    jmp     short pix_op_dec_4
6167      <1>
6168      pix_op_dec_1:
6169 0000E115 803D[799D0100]18 <1>    cmp     byte [v_bpp], 24 ; 24bpp
6170 0000E11C 7710        <1>    ja      short pix_op_dec_3 ; 32bpp
6171 0000E11E 7207        <1>    jb      short pix_op_dec_2 ; 16bpp
6172      <1>
6173      ; 24 bit true colors
6174 0000E120 E87C040000    <1>    call    pix_op_dec_24
6175 0000E125 EB0C        <1>    jmp     short pix_op_dec_4
6176      <1>
6177      ; 65536 colors (16bpp)
6178      pix_op_dec_2:
6179 0000E127 E868040000    <1>    call    pix_op_dec_16
6180 0000E12C EB05        <1>    jmp     short pix_op_dec_4
6181      <1>
6182      ; 32 bit true colors
6183      pix_op_dec_3:
6184 0000E12E E881040000    <1>    call    pix_op_dec_32
6185      pix_op_dec_4:
6186 0000E133 29F7        <1>    sub     edi, esi
6187 0000E135 893D[548E0100] <1>    mov     [u.r0], edi
6188      pix_op_dec_5:
6189 0000E13B C3          <1>    retn
6190      <1>
6191      pix_op_dec_w:
6192      ; 31/01/2021
6193      ; ecx = win start pos (row, column)
6194      ; edx = size (rows, columns)
6195 0000E13C E88AF6FFFF    <1>    call    sysvideo_15_12 ; window preparations
6196 0000E141 72F8        <1>    jc      short pix_op_dec_5
6197      <1>
6198 0000E143 F605[789D0100]20 <1>    test    byte [v_ops], 20h ; masked decrease color ?
6199 0000E14A 7405        <1>    jz      short pix_op_dec_w_0 ; no
6200 0000E14C E973100000    <1>    jmp     m_pix_op_dec_w
6201      ; window 'dec' color except mask color
6202      pix_op_dec_w_0:
6203      ; ecx = bytes per row (to be applied)
6204      ; edx = screen width in bytes
6205      ; ebx = row count
6206      <1>
6207 0000E151 8B3D[829D0100] <1>    mov     edi, [v_str]
6208      <1>
6209 0000E157 803D[799D0100]08 <1>    cmp     byte [v_bpp], 8 ; 8bpp
6210 0000E15E 7707        <1>    ja      short pix_op_dec_w_1
6211      <1>
6212      ; 256 colors (8bpp)
6213 0000E160 BD[8AE50000]   <1>    mov     ebp, pix_op_dec_8
6214 0000E165 EB1E        <1>    jmp     short pix_op_dec_w_4
6215      <1>
6216      pix_op_dec_w_1:
6217 0000E167 803D[799D0100]18 <1>    cmp     byte [v_bpp], 24 ; 24bpp
6218 0000E16E 7710        <1>    ja      short pix_op_dec_w_3 ; 32bpp
6219 0000E170 7207        <1>    jb      short pix_op_dec_w_2 ; 16bpp
6220      <1>
6221      ; 24 bit true colors
6222 0000E172 BD[A1E50000]   <1>    mov     ebp, pix_op_dec_24
6223 0000E177 EB0C        <1>    jmp     short pix_op_dec_w_4
6224      <1>
6225      ; 65536 colors (16bpp)
6226      pix_op_dec_w_2:
6227 0000E179 BD[94E50000]   <1>    mov     ebp, pix_op_dec_16
6228 0000E17E EB05        <1>    jmp     short pix_op_dec_w_4
6229      <1>
6230      ; 32 bit true colors
6231      pix_op_dec_w_3:
6232 0000E180 BD[B4E50000]   <1>    mov     ebp, pix_op_dec_32
6233      pix_op_dec_w_4:
6234 0000E185 E940FDFFFF    <1>    jmp     pix_op_dec_w_x
6235      <1>
6236      pix_op_blk:
6237      ; 11/08/2022 (TRDOS 386 Kernel v2.0.5)
6238      ; 23/01/2021
6239      ; 22/02/2021
6240      ; 02/02/2021
6241      ; COPY PIXEL BLOCK -system to system-
6242      ; WRITE PIXEL BLOCKS -user to system-
6243      ;
6244      ; INPUT:
6245      ; -If BL bit 5 is 0-
6246      ; ECX = start position (row, column) (*)
6247      ; (HW = row, CX = column)
6248      ; EDX = size (rows, columns) (*)
6249      ; (HW = rows, DX = columns)
6250      ; (0 -> invalid)
6251      ; (1 -> horizontal or vertical line)
6252      ; ESI = destination (row, column) (***)
6253      ; -If BL bit 5 is 1-
6254      ; CL = color (8 bit, 256 colors)
6255      ; ECX = color (16 bit and true colors)
6256      ; EDX = count of blocks (not bytes)
6257      ; (limit: 2048 blocks/windows)
6258      ; ESI = user's buffer address
6259      ; contains 64 bit block data
6260      ; BLOCK ADDRESS - (row, col), dword
6261      ; (first 32 bits)
6262      ; BLOCK SIZE - (rows, cols), dword
6263      ; (second 32 bits)
6264      ; OUTPUT:
6265      ; [u.r0] will be > 0 if succesful
6266      <1>
6267      ; window option ([v_ops] bit 4) will be ignored
6268      ; (Function is used for display page coordinates)
6269      <1>
6270 0000E18A F605[789D0100]20 <1>    test    byte [v_ops], 20h ; masked or direct ?
6271 0000E191 755A        <1>    jnz     short pix_op_blk_u ; blocks from user's buffer
6272      <1>
6273 0000E193 89F0        <1>    mov     eax, esi ; destination position (row, col)
6274 0000E195 E8DEF6FFFF    <1>    call    calc_pixel_offset
6275 0000E19A 3B05[7E9D0100] <1>    cmp     eax, [v_siz]
6276 0000E1A0 734A        <1>    jnb     short pix_op_blk_retn ; out of display page
6277 0000E1A2 89C6        <1>    mov     esi, eax
6278 0000E1A4 E8ACF6FFFF    <1>    call    pixels_to_byte_count
6279 0000E1A9 89C7        <1>    mov     edi, eax
6280 0000E1AB 89D0        <1>    mov     eax, edx ; size
6281 0000E1AD E8C6F6FFFF    <1>    call    calc_pixel_offset
6282      ; 22/02/2021
6283 0000E1B2 3B05[7E9D0100] <1>    cmp     eax, [v_siz]
6284 0000E1B8 7732        <1>    ja      short pix_op_blk_retn ; out of display page
6285 0000E1BA 01C6        <1>    add     esi, eax
6286 0000E1BC 3B35[7E9D0100] <1>    cmp     esi, [v_siz]
6287 0000E1C2 7728        <1>    ja      short pix_op_blk_retn ; out of display page

```

```

6288 <1>
6289 0000E1C4 033D[7A9D0100] <1> add edi, [v_mem] ; destination address
6290 <1>
6291 <1> ; 23/01/2021
6292 <1> ;call pixels_to_byte_count
6293 <1> ;add edi, eax
6294 <1> ;jc short pix_op_blk_retn ; out of display page
6295 <1> ;cmp edi, [v_end]
6296 <1> ;ja short pix_op_blk_retn ; out of display page
6297 <1> ;sub edi, eax
6298 <1>
6299 0000E1CA E8FCF5FFFF <1> call sysvideo_15_12 ; window preparations
6300 0000E1CF 721B <1> jc short pix_op_blk_retn ; something wrong !?
6301 <1> ; ecx = bytes per row (to be applied)
6302 <1> ; edx = screen width in bytes
6303 <1> ; ebx = row count
6304 <1>
6305 0000E1D1 8B35[829D0100] <1> mov esi, [v_str] ; source address
6306 <1>
6307 <1> ; Note:
6308 <1> ; ecx & edx are already adjusted for pixel sizes
6309 <1> ; so, following code is proper all pixel sizes
6310 <1>
6311 0000E1D7 29CA <1> sub edx, ecx ; screen width - window width
6312 <1> pix_op_blk_0:
6313 0000E1D9 89C8 <1> mov eax, ecx
6314 0000E1DB 0105[548E0100] <1> add [u.r0], eax
6315 0000E1E1 F3A4 <1> rep movsb
6316 0000E1E3 89C1 <1> mov ecx, eax
6317 0000E1E5 01D6 <1> add esi, edx ; next row
6318 0000E1E7 01D7 <1> add edi, edx ; next row
6319 0000E1E9 4B <1> dec ebx
6320 0000E1EA 75ED <1> jnz short pix_op_blk_0
6321 <1> pix_op_blk_retn:
6322 0000E1EC C3 <1> ret
6323 <1>
6324 <1> pix_op_blk_u:
6325 <1> ; fill blocks (windows) with desired color
6326 <1> ; according to block definitions in user's buffer
6327 0000E1ED 81FA00080000 <1> cmp edx, 2048
6328 0000E1F3 7605 <1> jna short pix_op_blk_u_0
6329 <1> ; Maximum 2048 blocks
6330 0000E1F5 BA00080000 <1> mov edx, 2048
6331 <1> pix_op_blk_u_0:
6332 0000E1FA 8025[789D0100]DF <1> and byte [v_ops], ~20h ; clear masked bit
6333 0000E201 890D[8A9D0100] <1> mov [maskcolor], ecx ; save pixel color
6334 <1> ; 22/02/2021
6335 <1> ;mov ebp, edx ; save blocks count
6336 <1> ;push ebp
6337 <1> pix_op_blk_u_next:
6338 0000E207 52 <1> push edx
6339 <1> ;mov ecx, 8
6340 <1> ; 11/08/2022
6341 0000E208 29C9 <1> sub ecx, ecx
6342 0000E20A B108 <1> mov cl, 8
6343 0000E20C BF[929D0100] <1> mov edi, buffer8 ; 8 bytes small buffer
6344 <1> ; esi = user's buffer address
6345 0000E211 E83F290000 <1> call transfer_from_user_buffer
6346 0000E216 72D4 <1> jc short pix_op_blk_retn
6347 0000E218 01CE <1> add esi, ecx ; 22/02/2021
6348 0000E21A 56 <1> push esi
6349 0000E21B 8B15[929D0100] <1> mov edx, [buffer8] ; block start pos (row,col)
6350 0000E221 8B35[969D0100] <1> mov esi, [buffer8+4] ; block size (rows,cols)
6351 0000E227 8B0D[8A9D0100] <1> mov ecx, [maskcolor]
6352 0000E22D E849FCFFFF <1> call pix_op_new_w ; new (change) color (window)
6353 0000E232 5E <1> pop esi
6354 <1> ;pop ebp
6355 <1> ;dec ebp
6356 0000E233 5A <1> pop edx
6357 0000E234 4A <1> dec edx
6358 0000E235 75D0 <1> jnz short pix_op_blk_u_next
6359 0000E237 C3 <1> ret
6360 <1>
6361 <1> pix_op_lin:
6362 <1> ; 11/08/2022
6363 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
6364 <1> ; 12/02/2021
6365 <1> ; 11/02/2021
6366 <1> ; 10/02/2021
6367 <1> ; 05/02/2021
6368 <1> ; 02/02/2021
6369 <1> ; WRITE LINE -direct-
6370 <1> ; WRITE LINE(S) -via user's buffer-
6371 <1>
6372 <1> ; INPUT:
6373 <1> ; -If BL bit 5 is 0-
6374 <1> ; CL = color (8 bit, 256 colors)
6375 <1> ; ECX = color (16 bit and true colors)
6376 <1> ; DX = low 12 bits - size (length)
6377 <1> ; high 4 bits - direction or type
6378 <1> ; 0 - Horizontal line
6379 <1> ; 1 - Vertical line
6380 <1> ; > 1 - undefined, invalid
6381 <1> ; ESI = start position (row, column)
6382 <1> ; (HW = row, SI = column)
6383 <1> ; -If BL bit 5 is 1-
6384 <1> ; CL = color (8 bit, 256 colors)
6385 <1> ; ECX = color (16 bit and true colors)
6386 <1> ; DX = number of lines (in user buffer)
6387 <1> ; (limit: 2048 lines)
6388 <1> ; ESI = user's buffer
6389 <1> ; contains 64 bit data for lines
6390 <1> ; START POINT: 32 bit (row, col)
6391 <1> ; LENGTH: 32 bit
6392 <1> ; high 16 bits - 0
6393 <1> ; bit 0-11 - length
6394 <1> ; bit 12-15 - type (length)
6395 <1> ; OUTPUT:
6396 <1> ; [u.r0] will be > 0 if succesful
6397 <1>
6398 <1> ; window option ([v_ops] bit 4) will be ignored
6399 <1> ; (Function is used for display page coordinates)
6400 <1>
6401 <1> ; 10/02/2021
6402 0000E238 F605[789D0100]20 <1> test byte [v_ops], 20h ; masked or direct ?
6403 0000E23F 7444 <1> jz short pix_op_lin_vh ; direct (v/h lines)
6404 <1>
6405 <1> ; lines from user's buffer
6406 <1> pix_op_lin_u:
6407 <1> ; draw lines with desired color
6408 <1> ; according to line definitions in user's buffer
6409 0000E241 81FA00080000 <1> cmp edx, 2048
6410 0000E247 7605 <1> jna short pix_op_lin_u_0
6411 <1> ; Maximum 2048 lines

```

```

6412 0000E249 BA00080000      <1>      mov     edx, 2048
6413                                <1> pix_op_lin_u_0:
6414 0000E24E 890D[8A9D0100]      <1>      mov     [maskcolor], ecx ; save pixel color
6415 0000E254 89D5              <1>      mov     ebp, edx ; save line count
6416                                <1> pix_op_lin_u_next:
6417                                <1>      ;mov     ecx, 8
6418                                <1>      ; 11/08/2022
6419 0000E256 29C9              <1>      sub     ecx, ecx
6420 0000E258 B108              <1>      mov     cl, 8
6421 0000E25A BF[929D0100]      <1>      mov     edi, buffer8 ; 8 bytes small buffer
6422                                <1>      ; esi = user's buffer address
6423 0000E25F E8F1280000      <1>      call    transfer_from_user_buffer
6424 0000E264 721E              <1>      jc      short pix_op_lin_retn
6425 0000E266 01CE              <1>      add     esi, ecx ; 11/02/2021
6426 0000E268 56              <1>      push    esi
6427 0000E269 8B35[929D0100]      <1>      mov     esi, [buffer8] ; line start pos (row,col)
6428 0000E26F 8B15[969D0100]      <1>      mov     edx, [buffer8+4] ; line length
6429 0000E275 8B0D[8A9D0100]      <1>      mov     ecx, [maskcolor]
6430 0000E27B E805000000      <1>      call    pix_op_lin_vh ; new (change) color (window)
6431 0000E280 5E              <1>      pop     esi
6432 0000E281 4D              <1>      dec     ebp
6433 0000E282 75D2              <1>      jnz     short pix_op_lin_u_next
6434                                <1> pix_op_lin_retn:
6435 0000E284 C3              <1>      retn
6436                                <1>
6437                                <1> pix_op_lin_vh:
6438 0000E285 81FA38140000      <1>      cmp     edx, 1438h ; 1920*1080 (780hx438h) limit
6439 0000E28B 7762              <1>      ja      short pix_op_lin_err1 ; invalid type
6440                                <1>      ; (for current version)
6441 0000E28D 66F7C2FF0F      <1>      test    dx, 0FFFh
6442 0000E292 745B              <1>      jz      short pix_op_lin_err1 ; zero length!
6443                                <1>
6444 0000E294 89F0              <1>      mov     eax, esi ; start point (row, col)
6445 0000E296 E8DDF5FFFF      <1>      call    calc_pixel_offset
6446 0000E29B 3B05[7E9D0100]      <1>      cmp     eax, [v_siz]
6447 0000E2A1 734C              <1>      jnb     short pix_op_lin_err1 ; out of display page!
6448 0000E2A3 E8ADF5FFFF      <1>      call    pixels_to_byte_count
6449 0000E2A8 89C7              <1>      mov     edi, eax ; start point offset
6450 0000E2AA 033D[7A9D0100]      <1>      add     edi, [v_mem] ; LFB start address
6451 0000E2B0 89C8              <1>      mov     eax, ecx ; color
6452                                <1>
6453 0000E2B2 F6C610              <1>      test    dh, 10h
6454                                <1>      ;jz      pix_op_lin_h ; Horizontal line
6455                                <1>      ; 23/07/2022
6456 0000E2B5 7505              <1>      jnz     short pix_op_lin_v
6457 0000E2B7 E98A000000      <1>      jmp     pix_op_lin_h
6458                                <1>
6459                                <1> pix_op_lin_v:
6460                                <1>      ; Vertical line
6461 0000E2BC 80E60F              <1>      and     dh, 0Fh ; low 12 bits
6462 0000E2BF 51              <1>      push    ecx ; color
6463 0000E2C0 89D1              <1>      mov     ecx, edx
6464 0000E2C2 0FB705[769D0100]      <1>      movzx    eax, word [v_width]
6465 0000E2C9 89C3              <1>      mov     ebx, eax
6466                                <1>      ; 12/02/2021
6467 0000E2CB F7E2              <1>      mul     edx ; rows * [v_width]
6468 0000E2CD 01F8              <1>      add     eax, edi
6469 0000E2CF 3B05[869D0100]      <1>      cmp     eax, [v_end]
6470 0000E2D5 58              <1>      pop     eax ; color
6471 0000E2D6 7717              <1>      ja      short pix_op_lin_err1 ; out of display page
6472                                <1>      ; ecx = rows
6473 0000E2D8 89CA              <1>      mov     edx, ecx
6474                                <1>
6475 0000E2DA 803D[799D0100]08      <1>      cmp     byte [v_bpp], 8 ; 8bpp
6476 0000E2E1 770D              <1>      ja      short pix_op_lin_v_2
6477                                <1>      ; 256 colors (1 byte per pixel)
6478 0000E2E3 010D[548E0100]      <1>      add     [u.r0], ecx ; byte count
6479                                <1> pix_op_lin_v_1:
6480 0000E2E9 8807              <1>      mov     [edi], al
6481 0000E2EB 01DF              <1>      add     edi, ebx ; next row
6482 0000E2ED E2FA              <1>      loop    pix_op_lin_v_1
6483                                <1> pix_op_lin_err1:
6484 0000E2EF C3              <1>      retn
6485                                <1>
6486                                <1> pix_op_lin_v_2:
6487 0000E2F0 803D[799D0100]18      <1>      cmp     byte [v_bpp], 24 ; 24bpp
6488 0000E2F7 773A              <1>      ja      short pix_op_lin_v_6 ; 32bpp
6489 0000E2F9 7226              <1>      jb      short pix_op_lin_v_4 ; 16bpp
6490                                <1>
6491                                <1>      ; 24 bit true colors
6492                                <1>      ; * 3
6493 0000E2FB 53              <1>      push    ebx ; screen width in pixels
6494 0000E2FC D1E3              <1>      shl     ebx, 1
6495 0000E2FE 011C24              <1>      add     [esp], ebx
6496 0000E301 5B              <1>      pop     ebx ; screen width in bytes
6497 0000E302 010D[548E0100]      <1>      add     [u.r0], ecx
6498 0000E308 D1E2              <1>      shl     edx, 1
6499 0000E30A 0115[548E0100]      <1>      add     [u.r0], edx ; byte count
6500                                <1> pix_op_lin_v_3:
6501 0000E310 668907              <1>      mov     [edi], ax
6502 0000E313 C1C810              <1>      ror     eax, 16
6503 0000E316 884702              <1>      mov     [edi+2], al
6504 0000E319 C1C010              <1>      rol     eax, 16
6505 0000E31C 01DF              <1>      add     edi, ebx ; next row
6506 0000E31E E2F0              <1>      loop    pix_op_lin_v_3
6507 0000E320 C3              <1>      retn
6508                                <1>
6509                                <1> pix_op_lin_v_4:
6510                                <1>      ; 16 bit (65536) colors
6511 0000E321 D1E3              <1>      shl     ebx, 1
6512 0000E323 D1E2              <1>      shl     edx, 1
6513 0000E325 0115[548E0100]      <1>      add     [u.r0], edx
6514                                <1> pix_op_lin_v_5:
6515 0000E32B 668907              <1>      mov     [edi], ax
6516 0000E32E 01DF              <1>      add     edi, ebx ; next row
6517 0000E330 E2F9              <1>      loop    pix_op_lin_v_5
6518 0000E332 C3              <1>      retn
6519                                <1>
6520                                <1> pix_op_lin_v_6:
6521                                <1>      ; 32 bit true colors
6522 0000E333 C1E302              <1>      shl     ebx, 2
6523 0000E336 C1E202              <1>      shl     edx, 2
6524 0000E339 0115[548E0100]      <1>      add     [u.r0], edx ; byte count
6525                                <1> pix_op_lin_v_7:
6526 0000E33F 8907              <1>      mov     [edi], eax
6527 0000E341 01DF              <1>      add     edi, ebx ; next row
6528 0000E343 E2FA              <1>      loop    pix_op_lin_v_7
6529 0000E345 C3              <1>      retn
6530                                <1>
6531                                <1> pix_op_lin_h:
6532                                <1>      ; Horizontal line
6533 0000E346 80E60F              <1>      and     dh, 0Fh ; low 12 bits
6534 0000E349 89D1              <1>      mov     ecx, edx
6535 0000E34B 6601D6              <1>      add     si, dx ; start column + columns

```

```

6536 0000E34E 663B35[769D0100] <1>    cmp     si, [v_width] ; screen width
6537 0000E355 7711 <1>    ja      short pix_op_lin_err2 ; out of columns limit
6538 <1>
6539 0000E357 803D[799D0100]08 <1>    cmp     byte [v_bpp], 8 ; 8bpp
6540 0000E35E 7709 <1>    ja      short pix_op_lin_h_1
6541 <1>    ; 256 colors (1 byte per pixel)
6542 0000E360 010D[548E0100] <1>    add     [u.r0], ecx
6543 0000E366 F3AA <1>    rep     stosb
6544 <1> pix_op_lin_err2:
6545 0000E368 C3 <1>    retn
6546 <1>
6547 <1> pix_op_lin_h_1:
6548 0000E369 803D[799D0100]18 <1>    cmp     byte [v_bpp], 24 ; 24bpp
6549 0000E370 7728 <1>    ja      short pix_op_lin_h_4 ; 32bpp
6550 0000E372 721A <1>    jb      short pix_op_lin_h_3 ; 16bpp
6551 <1>
6552 <1>    ; 24 bit true colors
6553 <1>    ; * 3
6554 0000E374 0115[548E0100] <1>    add     [u.r0], edx
6555 0000E37A D1E2 <1>    shl     edx, 1
6556 0000E37C 0115[548E0100] <1>    add     [u.r0], edx
6557 <1> pix_op_lin_h_2:
6558 0000E382 66AB <1>    stosw
6559 0000E384 C1C810 <1>    ror     eax, 16
6560 0000E387 AA <1>    stosb
6561 0000E388 C1C010 <1>    rol     eax, 16
6562 0000E38B E2F5 <1>    loop    pix_op_lin_h_2
6563 0000E38D C3 <1>    retn
6564 <1>
6565 <1> pix_op_lin_h_3:
6566 <1>    ; 16 bit (65536) colors
6567 0000E38E D1E2 <1>    shl     edx, 1
6568 0000E390 0115[548E0100] <1>    add     [u.r0], edx
6569 0000E396 F366AB <1>    rep     stosw
6570 0000E399 C3 <1>    retn
6571 <1>
6572 <1> pix_op_lin_h_4:
6573 <1>    ; 32 bit true colors
6574 0000E39A C1E202 <1>    shl     edx, 2
6575 0000E39D 0115[548E0100] <1>    add     [u.r0], edx
6576 0000E3A3 F3AB <1>    rep     stosd
6577 0000E3A5 C3 <1>    retn
6578 <1>
6579 <1> pix_op_new_8:
6580 <1>    ; 8 bit colors (256 colors)
6581 <1>    ; CHANGE PIXEL COLOR
6582 <1>    ; ecx = pixel count per row
6583 <1>    ; al = color
6584 <1>    ; edi = start pixel address
6585 <1>
6586 0000E3A6 F3AA <1>    rep     stosb
6587 0000E3A8 C3 <1>    retn
6588 <1>
6589 <1> pix_op_new_16:
6590 <1>    ; 16 bit colors (65536 colors)
6591 <1>    ; CHANGE PIXEL COLOR
6592 <1>    ; ecx = pixel count per row
6593 <1>    ; ax = color
6594 <1>    ; edi = start pixel address
6595 <1>
6596 0000E3A9 F366AB <1>    rep     stosw
6597 0000E3AC C3 <1>    retn
6598 <1>
6599 <1> pix_op_new_24:
6600 <1>    ; 24 bit true colors
6601 <1>    ; CHANGE PIXEL COLOR
6602 <1>    ; ecx = pixel count per row
6603 <1>    ; eax = color
6604 <1>    ; edi = start pixel address
6605 <1>
6606 0000E3AD 66AB <1>    stosw
6607 0000E3AF C1C810 <1>    ror     eax, 16
6608 0000E3B2 AA <1>    stosb
6609 0000E3B3 C1C010 <1>    rol     eax, 16
6610 0000E3B6 E2F5 <1>    loop    pix_op_new_24
6611 0000E3B8 C3 <1>    retn
6612 <1>
6613 <1> pix_op_new_32:
6614 <1>    ; 32 bit true colors
6615 <1>    ; CHANGE PIXEL COLOR
6616 <1>    ; ecx = pixel count per row
6617 <1>    ; eax = color
6618 <1>    ; edi = start pixel address
6619 <1>
6620 0000E3B9 F3AB <1>    rep     stosd
6621 0000E3BB C3 <1>    retn
6622 <1>
6623 <1> pix_op_add_8:
6624 <1>    ; 8 bit colors (256 colors)
6625 <1>    ; ADD PIXEL COLOR
6626 <1>    ; ecx = pixel count per row
6627 <1>    ; al = color
6628 <1>    ; edi = start pixel address
6629 <1>
6630 0000E3BC 88C4 <1>    mov     ah, al
6631 <1> pix_op_add_8_0:
6632 0000E3BE 0207 <1>    add     al, [edi]
6633 0000E3C0 7302 <1>    jnc     short pix_op_add_8_1
6634 0000E3C2 B0FF <1>    mov     al, 0FFh ; Max. value
6635 <1> pix_op_add_8_1:
6636 0000E3C4 AA <1>    stosb
6637 0000E3C5 88E0 <1>    mov     al, ah
6638 0000E3C7 E2F5 <1>    loop    pix_op_add_8_0
6639 0000E3C9 C3 <1>    retn
6640 <1>
6641 <1> pix_op_add_16:
6642 <1>    ; 16 bit colors (65536 colors)
6643 <1>    ; ADD PIXEL COLOR
6644 <1>    ; ecx = pixel count per row
6645 <1>    ; ax = color
6646 <1>    ; edi = start pixel address
6647 <1>
6648 0000E3CA 89C2 <1>    mov     edx, eax
6649 <1> pix_op_add_16_0:
6650 0000E3CC 660307 <1>    add     ax, [edi]
6651 0000E3CF 7304 <1>    jnc     short pix_op_add_16_1
6652 0000E3D1 66B8FFFF <1>    mov     ax, 0FFFFh ; Max. value
6653 <1> pix_op_add_16_1:
6654 0000E3D5 66AB <1>    stosw
6655 0000E3D7 89D0 <1>    mov     eax, edx
6656 0000E3D9 E2F1 <1>    loop    pix_op_add_16_0
6657 0000E3DB C3 <1>    retn
6658 <1>
6659 <1> pix_op_add_24:

```

```

6660      <1>      ; 24 bit true colors
6661      <1>      ; ADD PIXEL COLOR
6662      <1>      ; ecx = pixel count per row
6663      <1>      ; eax = color
6664      <1>      ; edi = start pixel address
6665      <1>
6666      0000E3DC 53      <1>      push     ebx
6667      0000E3DD BBFFFFFF00 <1>      mov      ebx, 0FFFFFFh
6668      <1>      ;and     eax, ebx ; 0FFFFFFh
6669      0000E3E2 89C2    <1>      mov      edx, eax
6670      <1>      pix_op_add_24_0:
6671      0000E3E4 8B07    <1>      mov      eax, [edi]
6672      0000E3E6 21D8    <1>      and      eax, ebx ; 0FFFFFFh
6673      0000E3E8 01D0    <1>      add      eax, edx
6674      0000E3EA 39D8    <1>      cmp      eax, ebx
6675      0000E3EC 7602    <1>      jna      short pix_op_add_24_1
6676      0000E3EE 89D8    <1>      mov      eax, ebx ; 0FFFFFFh ; Max. value
6677      <1>      pix_op_add_24_1:
6678      0000E3F0 66AB    <1>      stosw
6679      0000E3F2 C1E810 <1>      shr      eax, 16
6680      0000E3F5 AA      <1>      stosb
6681      0000E3F6 E2EC    <1>      loop     pix_op_add_24_0
6682      0000E3F8 89D0    <1>      mov      eax, edx
6683      0000E3FA 5B      <1>      pop      ebx
6684      0000E3FB C3      <1>      retn
6685      <1>
6686      <1>      pix_op_add_32:
6687      <1>      ; 32 bit true colors
6688      <1>      ; ADD PIXEL COLOR
6689      <1>      ; ecx = pixel count per row
6690      <1>      ; eax = color
6691      <1>      ; edi = start pixel address
6692      <1>
6693      0000E3FC 89C2    <1>      mov      edx, eax
6694      <1>      pix_op_add_32_0:
6695      0000E3FE 0307    <1>      add      eax, [edi]
6696      0000E400 7303    <1>      jnc      short pix_op_add_32_1
6697      <1>      ;mov     eax, 0FFFFFFFh ; Max. value
6698      0000E402 29C0    <1>      sub      eax, eax
6699      0000E404 48      <1>      dec      eax
6700      <1>      pix_op_add_32_1:
6701      0000E405 AB      <1>      stosd
6702      0000E406 89D0    <1>      mov      eax, edx
6703      0000E408 E2F4    <1>      loop     pix_op_add_32_0
6704      0000E40A C3      <1>      retn
6705      <1>
6706      <1>      pix_op_sub_8:
6707      <1>      ; 8 bit colors (256 colors)
6708      <1>      ; SUBTRACT PIXEL COLOR
6709      <1>      ; ecx = pixel count per row
6710      <1>      ; al = color
6711      <1>      ; edi = start pixel address
6712      <1>
6713      0000E40B 88C4    <1>      mov      ah, al
6714      <1>      pix_op_sub_8_0:
6715      0000E40D 8A07    <1>      mov      al, [edi]
6716      0000E40F 28E0    <1>      sub      al, ah
6717      0000E411 7302    <1>      jnb      short pix_op_sub_8_1
6718      0000E413 30C0    <1>      xor      al, al ; 0 ; Min. value
6719      <1>      pix_op_sub_8_1:
6720      0000E415 AA      <1>      stosb
6721      0000E416 E2F5    <1>      loop     pix_op_sub_8_0
6722      0000E418 88E0    <1>      mov      al, ah
6723      0000E41A C3      <1>      retn
6724      <1>
6725      <1>      pix_op_sub_16:
6726      <1>      ; 16 bit colors (65536 colors)
6727      <1>      ; SUBTRACT PIXEL COLOR
6728      <1>      ; ecx = pixel count per row
6729      <1>      ; ax = color
6730      <1>      ; edi = start pixel address
6731      <1>
6732      0000E41B 89C2    <1>      mov      edx, eax
6733      <1>      pix_op_sub_16_0:
6734      0000E41D 66B07    <1>      mov      ax, [edi]
6735      0000E420 6629D0    <1>      sub      ax, dx
6736      0000E423 7302    <1>      jnb      short pix_op_sub_16_1
6737      0000E425 31C0    <1>      xor      eax, eax ; 0 ; Min. value
6738      <1>      pix_op_sub_16_1:
6739      0000E427 66AB    <1>      stosw
6740      0000E429 E2F2    <1>      loop     pix_op_sub_16_0
6741      0000E42B 89D0    <1>      mov      eax, edx
6742      0000E42D C3      <1>      retn
6743      <1>
6744      <1>      pix_op_sub_24:
6745      <1>      ; 24 bit true colors
6746      <1>      ; SUBTRACT PIXEL COLOR
6747      <1>      ; ecx = pixel count per row
6748      <1>      ; eax = color
6749      <1>      ; edi = start pixel address
6750      <1>
6751      <1>      ;and     eax, 0FFFFFFh
6752      0000E42E 89C2    <1>      mov      edx, eax
6753      <1>      pix_op_sub_24_0:
6754      0000E430 8B07    <1>      mov      eax, [edi]
6755      <1>      ; 27/02/2021
6756      0000E432 25FFFFFF00 <1>      and      eax, 0FFFFFFh
6757      0000E437 29D0    <1>      sub      eax, edx
6758      0000E439 7302    <1>      jnb      short pix_op_sub_24_1
6759      0000E43B 31C0    <1>      xor      eax, eax ; 0 ; Min. value
6760      <1>      pix_op_sub_24_1:
6761      0000E43D 66AB    <1>      stosw
6762      0000E43F C1E810 <1>      shr      eax, 16
6763      0000E442 AA      <1>      stosb
6764      0000E443 E2EB    <1>      loop     pix_op_sub_24_0
6765      0000E445 89D0    <1>      mov      eax, edx
6766      0000E447 C3      <1>      retn
6767      <1>
6768      <1>      pix_op_sub_32:
6769      <1>      ; 32 bit true colors
6770      <1>      ; SUBTRACT PIXEL COLOR
6771      <1>      ; ecx = pixel count per row
6772      <1>      ; eax = color
6773      <1>      ; edi = start pixel address
6774      <1>
6775      0000E448 89C2    <1>      mov      edx, eax
6776      <1>      pix_op_sub_32_0:
6777      0000E44A 8B07    <1>      mov      eax, [edi]
6778      0000E44C 29D0    <1>      sub      eax, edx
6779      0000E44E 7302    <1>      jnb      short pix_op_sub_32_1
6780      0000E450 31C0    <1>      xor      eax, eax ; 0 ; Min. value
6781      <1>      pix_op_sub_32_1:
6782      0000E452 AB      <1>      stosd
6783      0000E453 E2F5    <1>      loop     pix_op_sub_32_0

```

```

6784 0000E455 89D0      <1>      mov     eax, edx
6785 0000E457 C3        <1>      retn
6786                    <1>
6787                    <1> pix_op_or_8:
6788                    <1>      ; 8 bit colors (256 colors)
6789                    <1>      ; OR PIXEL COLOR
6790                    <1>      ; ecx = pixel count per row
6791                    <1>      ; al = color
6792                    <1>      ; edi = start pixel address
6793                    <1>
6794                    <1> pix_op_or_8_0:
6795 0000E458 0807      <1>      or      [edi], al
6796 0000E45A 47        <1>      inc     edi
6797 0000E45B E2FB      <1>      loop    pix_op_or_8_0
6798 0000E45D C3        <1>      retn
6799                    <1>
6800                    <1> pix_op_or_16:
6801                    <1>      ; 16 bit colors (65536 colors)
6802                    <1>      ; OR PIXEL COLOR
6803                    <1>      ; ecx = pixel count per row
6804                    <1>      ; ax = color
6805                    <1>      ; edi = start pixel address
6806                    <1>
6807                    <1> pix_op_or_16_0:
6808 0000E45E 660907    <1>      or      [edi], ax
6809 0000E461 47        <1>      inc     edi
6810 0000E462 47        <1>      inc     edi
6811 0000E463 E2F9      <1>      loop    pix_op_or_16_0
6812 0000E465 C3        <1>      retn
6813                    <1>
6814                    <1> pix_op_or_24:
6815                    <1>      ; 24 bit true colors
6816                    <1>      ; OR PIXEL COLOR
6817                    <1>      ; ecx = pixel count per row
6818                    <1>      ; eax = color
6819                    <1>      ; edi = start pixel address
6820                    <1>
6821 0000E466 89C2      <1>      mov     edx, eax
6822                    <1> pix_op_or_24_0:
6823 0000E468 0B07      <1>      or      eax, [edi]
6824 0000E46A 66AB      <1>      stosw
6825 0000E46C C1E810    <1>      shr     eax, 16
6826 0000E46F AA        <1>      stosb
6827 0000E470 89D0      <1>      mov     eax, edx
6828 0000E472 E2F4      <1>      loop    pix_op_or_24_0
6829 0000E474 C3        <1>      retn
6830                    <1>
6831                    <1> pix_op_or_32:
6832                    <1>      ; 32 bit true colors
6833                    <1>      ; OR PIXEL COLOR
6834                    <1>      ; ecx = pixel count per row
6835                    <1>      ; eax = color
6836                    <1>      ; edi = start pixel address
6837                    <1>
6838                    <1>      ;mov     edx, eax
6839                    <1> pix_op_or_32_0:
6840                    <1>      ;or      eax, [edi]
6841                    <1>      ;stosd
6842                    <1>      ;mov     eax, edx
6843 0000E475 0907      <1>      or      [edi], eax
6844 0000E477 83C704    <1>      add     edi, 4
6845 0000E47A E2F9      <1>      loop    pix_op_or_32_0
6846 0000E47C C3        <1>      retn
6847                    <1>
6848                    <1> pix_op_and_8:
6849                    <1>      ; 8 bit colors (256 colors)
6850                    <1>      ; AND PIXEL COLOR
6851                    <1>      ; ecx = pixel count per row
6852                    <1>      ; al = color
6853                    <1>      ; edi = start pixel address
6854                    <1>
6855                    <1> pix_op_and_8_0:
6856 0000E47D 2007      <1>      and     [edi], al
6857 0000E47F 47        <1>      inc     edi
6858 0000E480 E2FB      <1>      loop    pix_op_and_8_0
6859 0000E482 C3        <1>      retn
6860                    <1>
6861                    <1> pix_op_and_16:
6862                    <1>      ; 16 bit colors (65536 colors)
6863                    <1>      ; AND PIXEL COLOR
6864                    <1>      ; ecx = pixel count per row
6865                    <1>      ; ax = color
6866                    <1>      ; edi = start pixel address
6867                    <1>
6868                    <1> pix_op_and_16_0:
6869 0000E483 662107    <1>      and     [edi], ax
6870 0000E486 47        <1>      inc     edi
6871 0000E487 47        <1>      inc     edi
6872 0000E488 E2F9      <1>      loop    pix_op_and_16_0
6873 0000E48A C3        <1>      retn
6874                    <1>
6875                    <1> pix_op_and_24:
6876                    <1>      ; 24 bit true colors
6877                    <1>      ; AND PIXEL COLOR
6878                    <1>      ; ecx = pixel count per row
6879                    <1>      ; eax = color
6880                    <1>      ; edi = start pixel address
6881                    <1>
6882 0000E48B 89C2      <1>      mov     edx, eax
6883                    <1> pix_op_and_24_0:
6884 0000E48D 2307      <1>      and     eax, [edi]
6885 0000E48F 66AB      <1>      stosw
6886 0000E491 C1E810    <1>      shr     eax, 16
6887 0000E494 AA        <1>      stosb
6888 0000E495 89D0      <1>      mov     eax, edx
6889 0000E497 E2F4      <1>      loop    pix_op_and_24_0
6890 0000E499 C3        <1>      retn
6891                    <1>
6892                    <1> pix_op_and_32:
6893                    <1>      ; 32 bit true colors
6894                    <1>      ; AND PIXEL COLOR
6895                    <1>      ; ecx = pixel count per row
6896                    <1>      ; eax = color
6897                    <1>      ; edi = start pixel address
6898                    <1>
6899                    <1>      ;mov     edx, eax
6900                    <1> pix_op_and_32_0:
6901                    <1>      ;and     eax, [edi]
6902                    <1>      ;stosd
6903                    <1>      ;mov     eax, edx
6904 0000E49A 2107      <1>      and     [edi], eax
6905 0000E49C 83C704    <1>      add     edi, 4
6906 0000E49F E2F9      <1>      loop    pix_op_and_32_0
6907 0000E4A1 C3        <1>      retn

```

```

6908 <1>
6909 <1> pix_op_xor_8:
6910 <1> ; 8 bit colors (256 colors)
6911 <1> ; XOR PIXEL COLOR
6912 <1> ; ecx = pixel count per row
6913 <1> ; al = color
6914 <1> ; edi = start pixel address
6915 <1>
6916 <1> pix_op_xor_8_0:
6917 0000E4A2 3007 <1> xor [edi], al
6918 0000E4A4 47 <1> inc edi
6919 0000E4A5 E2FB <1> loop pix_op_xor_8_0
6920 0000E4A7 C3 <1> retn
6921 <1>
6922 <1> pix_op_xor_16:
6923 <1> ; 16 bit colors (65536 colors)
6924 <1> ; XOR PIXEL COLOR
6925 <1> ; ecx = pixel count per row
6926 <1> ; ax = color
6927 <1> ; edi = start pixel address
6928 <1>
6929 <1> pix_op_xor_16_0:
6930 0000E4A8 663107 <1> xor [edi], ax
6931 0000E4AB 47 <1> inc edi
6932 0000E4AC 47 <1> inc edi
6933 0000E4AD E2F9 <1> loop pix_op_xor_16_0
6934 0000E4AF C3 <1> retn
6935 <1>
6936 <1> pix_op_xor_24:
6937 <1> ; 24 bit true colors
6938 <1> ; XOR PIXEL COLOR
6939 <1> ; ecx = pixel count per row
6940 <1> ; eax = color
6941 <1> ; edi = start pixel address
6942 <1>
6943 0000E4B0 89C2 <1> mov edx, eax
6944 <1> pix_op_xor_24_0:
6945 0000E4B2 3307 <1> xor eax, [edi]
6946 0000E4B4 66AB <1> stosw
6947 0000E4B6 C1E810 <1> shr eax, 16
6948 0000E4B9 AA <1> stosb
6949 0000E4BA 89D0 <1> mov eax, edx
6950 0000E4BC E2F4 <1> loop pix_op_xor_24_0
6951 0000E4BE C3 <1> retn
6952 <1>
6953 <1> pix_op_xor_32:
6954 <1> ; 32 bit true colors
6955 <1> ; XOR PIXEL COLOR
6956 <1> ; ecx = pixel count per row
6957 <1> ; eax = color
6958 <1> ; edi = start pixel address
6959 <1>
6960 <1> ;mov edx, eax
6961 <1> pix_op_xor_32_0:
6962 <1> ;xor eax, [edi]
6963 <1> ;stosd
6964 <1> ;mov eax, edx
6965 0000E4BF 3107 <1> xor [edi], eax
6966 0000E4C1 83C704 <1> add edi, 4
6967 0000E4C4 E2F9 <1> loop pix_op_xor_32_0
6968 0000E4C6 C3 <1> retn
6969 <1>
6970 <1> pix_op_mix_8:
6971 <1> ; 8 bit colors (256 colors)
6972 <1> ; MIX (AVERAGE) PIXEL COLORS
6973 <1> ; ecx = pixel count per row
6974 <1> ; al = color
6975 <1> ; edi = start pixel address
6976 <1>
6977 0000E4C7 88C4 <1> mov ah, al
6978 <1> pix_op_mix_8_0:
6979 0000E4C9 0207 <1> add al, [edi]
6980 0000E4CB D0D8 <1> rcr al, 1
6981 0000E4CD AA <1> stosb
6982 0000E4CE 88E0 <1> mov al, ah
6983 0000E4D0 E2F7 <1> loop pix_op_mix_8_0
6984 0000E4D2 C3 <1> retn
6985 <1>
6986 <1> pix_op_mix_16:
6987 <1> ; 16 bit colors (65536 colors)
6988 <1> ; MIX (AVERAGE) PIXEL COLORS
6989 <1> ; ecx = pixel count per row
6990 <1> ; ax = color
6991 <1> ; edi = start pixel address
6992 <1>
6993 0000E4D3 89C2 <1> mov edx, eax
6994 <1> pix_op_mix_16_0:
6995 0000E4D5 660307 <1> add ax, [edi]
6996 0000E4D8 66D1D8 <1> rcr ax, 1
6997 0000E4DB 66AB <1> stosw
6998 0000E4DD 89D0 <1> mov eax, edx
6999 0000E4DF E2F4 <1> loop pix_op_mix_16_0
7000 0000E4E1 C3 <1> retn
7001 <1>
7002 <1> pix_op_mix_24:
7003 <1> ; 24 bit true colors
7004 <1> ; MIX (AVERAGE) PIXEL COLORS
7005 <1> ; ecx = pixel count per row
7006 <1> ; eax = color
7007 <1> ; edi = start pixel address
7008 <1>
7009 0000E4E2 53 <1> push ebx
7010 0000E4E3 BBFFFFFF00 <1> mov ebx, 0FFFFFFh
7011 <1> ;and eax, ebx ; 0FFFFFFh
7012 0000E4E8 89C2 <1> mov edx, eax
7013 <1> pix_op_mix_24_0:
7014 0000E4EA 8B07 <1> mov eax, [edi]
7015 0000E4EC 21D8 <1> and eax, ebx ; 0FFFFFFh
7016 0000E4EE 01D0 <1> add eax, edx
7017 0000E4F0 D1E8 <1> shr eax, 1
7018 <1> ;rcr eax, 1
7019 0000E4F2 66AB <1> stosw
7020 0000E4F4 C1E810 <1> shr eax, 16
7021 0000E4F7 AA <1> stosb
7022 0000E4F8 E2F0 <1> loop pix_op_mix_24_0
7023 0000E4FA 89D0 <1> mov eax, edx
7024 0000E4FC 5B <1> pop ebx
7025 0000E4FD C3 <1> retn
7026 <1>
7027 <1> pix_op_mix_32:
7028 <1> ; 32 bit true colors
7029 <1> ; MIX (AVERAGE) PIXEL COLORS
7030 <1> ; ecx = pixel count per row
7031 <1> ; eax = color

```



```

7032      <1>      ; edi = start pixel address
7033      <1>
7034      0000E4FE 89C2      <1>      mov     edx, eax
7035      <1>      pix_op_mix_32_0:
7036      0000E500 0307      <1>      add     eax, [edi]
7037      0000E502 D1D8      <1>      rcr     eax, 1
7038      0000E504 AB        <1>      stosd
7039      0000E505 89D0      <1>      mov     eax, edx
7040      0000E507 E2F7      <1>      loop    pix_op_mix_32_0
7041      0000E509 C3        <1>      retn
7042      <1>
7043      <1>      pix_op_not_8:
7044      <1>      ; 8 bit colors (256 colors)
7045      <1>      ; NOT PIXEL COLOR
7046      <1>      ; ecx = pixel count per row
7047      <1>      ; edi = start pixel address
7048      <1>
7049      <1>      pix_op_not_8_0:
7050      0000E50A F617      <1>      not     byte [edi]
7051      0000E50C 47        <1>      inc     edi
7052      0000E50D E2FB      <1>      loop    pix_op_not_8_0
7053      0000E50F C3        <1>      retn
7054      <1>
7055      <1>      pix_op_not_16:
7056      <1>      ; 16 bit colors (65536 colors)
7057      <1>      ; NOT PIXEL COLOR
7058      <1>      ; ecx = pixel count per row
7059      <1>      ; edi = start pixel address
7060      <1>
7061      <1>      pix_op_not_16_0:
7062      0000E510 66F717    <1>      not     word [edi]
7063      0000E513 47        <1>      inc     edi
7064      0000E514 47        <1>      inc     edi
7065      0000E515 E2F9      <1>      loop    pix_op_not_16_0
7066      0000E517 C3        <1>      retn
7067      <1>
7068      <1>      pix_op_not_24:
7069      <1>      ; 24 bit true colors
7070      <1>      ; NOT PIXEL COLOR
7071      <1>      ; ecx = pixel count per row
7072      <1>      ; edi = start pixel address
7073      <1>
7074      <1>      pix_op_not_24_0:
7075      0000E518 66F717    <1>      not     word [edi]
7076      0000E51B 47        <1>      inc     edi
7077      0000E51C 47        <1>      inc     edi
7078      0000E51D F617      <1>      not     byte [edi]
7079      0000E51F 47        <1>      inc     edi
7080      0000E520 E2F6      <1>      loop    pix_op_not_24_0
7081      0000E522 C3        <1>      retn
7082      <1>
7083      <1>      pix_op_not_32:
7084      <1>      ; 32 bit true colors
7085      <1>      ; NOT PIXEL COLOR
7086      <1>      ; ecx = pixel count per row
7087      <1>      ; eax = color
7088      <1>      ; edi = start pixel address
7089      <1>      pix_op_not_32_0:
7090      0000E523 F717      <1>      not     dword [edi]
7091      0000E525 83C704    <1>      add     edi, 4
7092      0000E528 E2F9      <1>      loop    pix_op_not_32_0
7093      0000E52A C3        <1>      retn
7094      <1>
7095      <1>      pix_op_neg_8:
7096      <1>      ; 8 bit colors (256 colors)
7097      <1>      ; NEG PIXEL COLOR
7098      <1>      ; ecx = pixel count per row
7099      <1>      ; edi = start pixel address
7100      <1>
7101      <1>      pix_op_neg_8_0:
7102      0000E52B F61F      <1>      neg     byte [edi]
7103      0000E52D 47        <1>      inc     edi
7104      0000E52E E2FB      <1>      loop    pix_op_neg_8_0
7105      0000E530 C3        <1>      retn
7106      <1>
7107      <1>      pix_op_neg_16:
7108      <1>      ; 16 bit colors (65536 colors)
7109      <1>      ; NEG PIXEL COLOR
7110      <1>      ; ecx = pixel count per row
7111      <1>      ; edi = start pixel address
7112      <1>
7113      <1>      pix_op_neg_16_0:
7114      0000E531 66F71F    <1>      neg     word [edi]
7115      0000E534 47        <1>      inc     edi
7116      0000E535 47        <1>      inc     edi
7117      0000E536 E2F9      <1>      loop    pix_op_neg_16_0
7118      0000E538 C3        <1>      retn
7119      <1>
7120      <1>      pix_op_neg_24:
7121      <1>      ; 24 bit true colors
7122      <1>      ; NEG PIXEL COLOR
7123      <1>      ; ecx = pixel count per row
7124      <1>      ; edi = start pixel address
7125      <1>
7126      <1>      pix_op_neg_24_0:
7127      0000E539 8B07      <1>      mov     eax, [edi]
7128      0000E53B 25FFFFFF00 <1>      and     eax, 0FFFFFFh
7129      0000E540 F7D8      <1>      neg     eax
7130      0000E542 66AB      <1>      stosw
7131      0000E544 C1E810    <1>      shr     eax, 16
7132      0000E547 AA        <1>      stosb
7133      0000E548 E2EF      <1>      loop    pix_op_neg_24_0
7134      0000E54A C3        <1>      retn
7135      <1>
7136      <1>      pix_op_neg_32:
7137      <1>      ; 32 bit true colors
7138      <1>      ; NEG PIXEL COLOR
7139      <1>      ; ecx = pixel count per row
7140      <1>      ; eax = color
7141      <1>      ; edi = start pixel address
7142      <1>      pix_op_neg_32_0:
7143      0000E54B F71F      <1>      neg     dword [edi]
7144      0000E54D 83C704    <1>      add     edi, 4
7145      0000E550 E2F9      <1>      loop    pix_op_neg_32_0
7146      0000E552 C3        <1>      retn
7147      <1>
7148      <1>      pix_op_inc_8:
7149      <1>      ; 8 bit colors (256 colors)
7150      <1>      ; INCREASE PIXEL COLOR
7151      <1>      ; ecx = pixel count per row
7152      <1>      ; edi = start pixel address
7153      <1>
7154      <1>      pix_op_inc_8_0:
7155      0000E553 FE07      <1>      inc     byte [edi]

```

```

7156 0000E555 7502      <1>      jnz      short pix_op_inc_8_1
7157                    <1>      ;mov      [edi], 0FFh ; Max. value
7158 0000E557 FE0F      <1>      dec      byte [edi]
7159                    <1>      pix_op_inc_8_1:
7160 0000E559 47          <1>      inc      edi
7161 0000E55A E2F7      <1>      loop     pix_op_inc_8_0
7162 0000E55C C3          <1>      retn
7163                    <1>
7164                    <1>      pix_op_inc_16:
7165                    <1>      ; 16 bit colors (65536 colors)
7166                    <1>      ; INCREASE PIXEL COLOR
7167                    <1>      ; ecx = pixel count per row
7168                    <1>      ; edi = start pixel address
7169                    <1>
7170                    <1>      pix_op_inc_16_0:
7171 0000E55D 66FF07     <1>      inc      word [edi]
7172 0000E560 7503      <1>      jnz      short pix_op_inc_16_1
7173                    <1>      ;mov      word [edi], 0FFFFh ; Max. value
7174 0000E562 66FF0F     <1>      dec      word [edi]
7175                    <1>      pix_op_inc_16_1:
7176 0000E565 47          <1>      inc      edi
7177 0000E566 47          <1>      inc      edi
7178 0000E567 E2F4      <1>      loop     pix_op_inc_16_0
7179 0000E569 C3          <1>      retn
7180                    <1>
7181                    <1>      pix_op_inc_24:
7182                    <1>      ; 24 bit true colors
7183                    <1>      ; INCREASE PIXEL COLOR
7184                    <1>      ; ecx = pixel count per row
7185                    <1>      ; edi = start pixel address
7186                    <1>
7187                    <1>      pix_op_inc_24_0:
7188 0000E56A 8B07      <1>      mov      eax, [edi]
7189 0000E56C 40          <1>      inc      eax
7190 0000E56D 25FFFFFF00 <1>      and      eax, 0FFFFFFh
7191 0000E572 7501      <1>      jnz      short pix_op_inc_24_1
7192                    <1>      ;mov      eax, 0FFFFFFh ; Max. value
7193 0000E574 48          <1>      dec      eax ; 0FFFFFFFh
7194                    <1>      pix_op_inc_24_1:
7195 0000E575 66AB      <1>      stosw
7196 0000E577 C1E810     <1>      shr      eax, 16
7197 0000E57A AA          <1>      stosb
7198 0000E57B E2ED      <1>      loop     pix_op_inc_24_0
7199 0000E57D C3          <1>      retn
7200                    <1>
7201                    <1>      pix_op_inc_32:
7202                    <1>      ; 32 bit true colors
7203                    <1>      ; INCREASE PIXEL COLOR
7204                    <1>      ; ecx = pixel count per row
7205                    <1>      ; edi = start pixel address
7206                    <1>
7207                    <1>      pix_op_inc_32_0:
7208 0000E57E FF07      <1>      inc      dword [edi]
7209 0000E580 7502      <1>      jnz      short pix_op_inc_32_1
7210                    <1>      ;mov      dword [edi], 0FFFFFFFh ; Max. value
7211 0000E582 FF0F      <1>      dec      dword [edi]
7212                    <1>      pix_op_inc_32_1:
7213 0000E584 83C704     <1>      add      edi, 4
7214 0000E587 E2F5      <1>      loop     pix_op_inc_32_0
7215 0000E589 C3          <1>      retn
7216                    <1>
7217                    <1>      pix_op_dec_8:
7218                    <1>      ; 8 bit colors (256 colors)
7219                    <1>      ; DECREASE PIXEL COLOR
7220                    <1>      ; ecx = pixel count per row
7221                    <1>      ; edi = start pixel address
7222                    <1>
7223                    <1>      pix_op_dec_8_0:
7224 0000E58A FE0F      <1>      dec      byte [edi]
7225 0000E58C 7902      <1>      jns      short pix_op_dec_8_1
7226 0000E58E FE07      <1>      inc      byte [edi] ; 0 ; Min. value
7227                    <1>      pix_op_dec_8_1:
7228 0000E590 47          <1>      inc      edi
7229 0000E591 E2F7      <1>      loop     pix_op_dec_8_0
7230 0000E593 C3          <1>      retn
7231                    <1>
7232                    <1>      pix_op_dec_16:
7233                    <1>      ; 16 bit colors (65536 colors)
7234                    <1>      ; DECREASE PIXEL COLOR
7235                    <1>      ; ecx = pixel count per row
7236                    <1>      ; edi = start pixel address
7237                    <1>
7238                    <1>      pix_op_dec_16_0:
7239 0000E594 66FF0F     <1>      dec      word [edi]
7240 0000E597 7903      <1>      jns      short pix_op_dec_16_1
7241 0000E599 66FF07     <1>      inc      word [edi] ; 0 ; Min. value
7242                    <1>      pix_op_dec_16_1:
7243 0000E59C 47          <1>      inc      edi
7244 0000E59D 47          <1>      inc      edi
7245 0000E59E E2F4      <1>      loop     pix_op_dec_16_0
7246 0000E5A0 C3          <1>      retn
7247                    <1>
7248                    <1>      pix_op_dec_24:
7249                    <1>      ; 24 bit true colors
7250                    <1>      ; DECREASE PIXEL COLOR
7251                    <1>      ; ecx = pixel count per row
7252                    <1>      ; edi = start pixel address
7253                    <1>
7254                    <1>      pix_op_dec_24_0:
7255 0000E5A1 8B07      <1>      mov      eax, [edi]
7256 0000E5A3 25FFFFFF00 <1>      and      eax, 0FFFFFFh
7257 0000E5A8 7401      <1>      jz       short pix_op_dec_24_1
7258                    <1>      ; 0 ; Min. value
7259 0000E5AA 48          <1>      dec      eax
7260                    <1>      pix_op_dec_24_1:
7261 0000E5AB 66AB      <1>      stosw
7262 0000E5AD C1E810     <1>      shr      eax, 16
7263 0000E5B0 AA          <1>      stosb
7264 0000E5B1 E2B7      <1>      loop     pix_op_inc_24_0
7265 0000E5B3 C3          <1>      retn
7266                    <1>
7267                    <1>      pix_op_dec_32:
7268                    <1>      ; 32 bit true colors
7269                    <1>      ; DECREASE PIXEL COLOR
7270                    <1>      ; ecx = pixel count per row
7271                    <1>      ; edi = start pixel address
7272                    <1>
7273                    <1>      pix_op_dec_32_0:
7274 0000E5B4 FF0F      <1>      dec      dword [edi]
7275 0000E5B6 7902      <1>      jns      short pix_op_dec_32_1
7276 0000E5B8 FF07      <1>      inc      dword [edi] ; 0 ; Min. value
7277                    <1>      pix_op_dec_32_1:
7278 0000E5BA 83C704     <1>      add      edi, 4
7279 0000E5BD E2F5      <1>      loop     pix_op_dec_32_0

```

```

7280 0000E5BF 89D0      <1>      mov     eax, edx
7281 0000E5C1 C3        <1>      retn
7282                  <1>
7283                  <1> pix_op_rpl_8:
7284                  <1>      ; 8 bit colors (256 colors)
7285                  <1>      ; REPLACE PIXEL COLORS
7286                  <1>      ; ecx = pixel count per row
7287                  <1>      ; al = new color
7288                  <1>      ; byte [maskcolor] = old color
7289                  <1>      ; edi = start pixel address
7290                  <1>
7291 0000E5C2 8A25[8A9D0100] <1>      mov     ah, [maskcolor]
7292                  <1> pix_op_rpl_8_0:
7293                  <1>      cmp     ah, [edi]
7294                  <1>      jne     short pix_op_rpl_8_1
7295                  <1>      mov     [edi], al
7296                  <1> pix_op_rpl_8_1:
7297                  <1>      inc     edi
7298                  <1>      loop   pix_op_rpl_8_0
7299                  <1>      retn
7300                  <1>
7301                  <1> pix_op_rpl_16:
7302                  <1>      ; 16 bit colors (65536 colors)
7303                  <1>      ; REPLACE PIXEL COLORS
7304                  <1>      ; ecx = pixel count per row
7305                  <1>      ; ax = new color
7306                  <1>      ; word [maskcolor] = old color
7307                  <1>      ; edi = start pixel address
7308                  <1>
7309 0000E5D2 8B15[8A9D0100] <1>      mov     edx, [maskcolor]
7310                  <1> pix_op_rpl_16_0:
7311                  <1>      cmp     dx, [edi]
7312                  <1>      jne     short pix_op_rpl_16_1
7313                  <1>      mov     [edi], ax
7314                  <1> pix_op_rpl_16_1:
7315                  <1>      inc     edi
7316                  <1>      inc     edi
7317                  <1>      loop   pix_op_rpl_16_0
7318                  <1>      retn
7319                  <1>
7320                  <1> pix_op_rpl_24:
7321                  <1>      ; 24 bit true colors
7322                  <1>      ; REPLACE PIXEL COLORS
7323                  <1>      ; ecx = pixel count per row
7324                  <1>      ; eax = new color
7325                  <1>      ; [maskcolor] = old color
7326                  <1>      ; edi = start pixel address
7327                  <1>
7328                  <1> pix_op_rpl_24_0:
7329                  <1>      mov     edx, [edi]
7330                  <1>      and     edx, 0FFFFFFh
7331                  <1>      cmp     edx, [maskcolor]
7332                  <1>      je      short pix_op_rpl_24_1
7333                  <1>      add     edi, 3
7334                  <1>      loop   pix_op_rpl_24_0
7335                  <1>      retn
7336                  <1> pix_op_rpl_24_1:
7337                  <1>      stosb
7338                  <1>      ror     eax, 8
7339                  <1>      stosw
7340                  <1>      rol     eax, 8
7341                  <1>      loop   pix_op_rpl_24_0
7342                  <1>      retn
7343                  <1>
7344                  <1> pix_op_rpl_32:
7345                  <1>      ; 32 bit true colors
7346                  <1>      ; REPLACE PIXEL COLORS
7347                  <1>      ; ecx = pixel count per row
7348                  <1>      ; eax = new color
7349                  <1>      ; [maskcolor] = old color
7350                  <1>      ; edi = start pixel address
7351                  <1>
7352 0000E607 8B15[8A9D0100] <1>      mov     edx, [maskcolor]
7353                  <1> pix_op_rpl_32_0:
7354                  <1>      cmp     edx, [edi]
7355                  <1>      jne     short pix_op_rpl_32_2
7356                  <1>      stosd
7357                  <1>      loop   pix_op_rpl_32_0
7358                  <1>      retn
7359                  <1> pix_op_rpl_32_2:
7360                  <1>      add     edi, 4
7361                  <1>      loop   pix_op_rpl_32_0
7362                  <1>      retn
7363                  <1>
7364                  <1> pix_op_chr:
7365                  <1>      ; 23/07/2022 - TRDOS 386 v2.0.5
7366                  <1>      ; 15/02/2021
7367                  <1>      ; 05/02/2021
7368                  <1>      ; WRITE CHARACTER (FONT)
7369                  <1>      ; 05/01/2021 ([ufont])
7370                  <1>      ; 01/01/2021
7371                  <1>      ; CL = char's color (8 bit, 256 colors)
7372                  <1>      ; ECX = char's color (16 bit and true colors)
7373                  <1>      ; DL = Character's ASCII code
7374                  <1>      ; DH bit 0 -> font height
7375                  <1>      ; 0 -> 8x16 character font
7376                  <1>      ; 1 -> 8x8 character font
7377                  <1>      ; DH bit 1 & 2 -> scale
7378                  <1>      ; 0 = 1/1 (8 pixels per char row)
7379                  <1>      ; 1 = 2/1 (16 pixels per char row)
7380                  <1>      ; 2 = 3/1 (24 pixels per char row)
7381                  <1>      ; 3 = 4/1 (32 pixels per char row)
7382                  <1>      ; DH bit 6 -> [ufont] option (1 = use [ufont])
7383                  <1>      ; If DH bit 7 = 1
7384                  <1>      ; USER FONT (from user buffer)
7385                  <1>      ; DL = 0 -> 8x8 (width: 1 byte per row)
7386                  <1>      ; DL = 1 -> 8x16
7387                  <1>      ; DL = 2 -> 16x16 (width: 2 bytes)
7388                  <1>      ; DL = 3 -> 16x32
7389                  <1>      ; DL = 4 -> 24x24 (width: 3 bytes)
7390                  <1>      ; DL = 5 -> 24x48
7391                  <1>      ; DL = 6 -> 32x32 (width: 4 bytes)
7392                  <1>      ; DL = 7 -> 32x64
7393                  <1>      ; DL > 7 -> invalid (unused)
7394                  <1>      ; EDI = user's font buffer address
7395                  <1>      ; (NOTE: byte order is as row0,row1,row2..)
7396                  <1>      ; ESI = start position (row, column) (*)
7397                  <1>      ; (HW = row, SI = column)
7398                  <1>
7399 0000E61B 89F0      <1>      mov     eax, esi ; start position
7400 0000E61D E856F2FFFF <1>      call    calc_pixel_offset
7401 0000E622 3B05[7E9D0100] <1>      cmp     eax, [v_siz]
7402 0000E628 736D      <1>      jnb     short pix_op_chr_err ; out of display page!
7403 0000E62A E826F2FFFF <1>      call    pixels_to_byte_count

```

```

7404      <1>      ; eax = font start offset
7405 0000E62F 0305[7A9D0100] <1>      add     eax, [v_mem] ; LFB start address
7406 0000E635 A3[829D0100] <1>      mov     [v_str], eax ; font start address
7407      <1>
7408 0000E63A 890D[8A9D0100] <1>      mov     [maskcolor], ecx ; save char's color
7409      <1>
7410 0000E640 8835[789D0100] <1>      mov     [v_ops], dh
7411      <1>
7412 0000E646 81E6FFFF0000 <1>      and     esi, 0FFFFh
7413 0000E64C 8935[929D0100] <1>      mov     [buffer8], esi ; start column
7414      <1>
7415 0000E652 31DB <1>      xor     ebx, ebx ; 0
7416 0000E654 31C0 <1>      xor     eax, eax ; 15/02/2021
7417      <1>
7418 0000E656 F6C680 <1>      test    dh, 80h
7419 0000E659 7577 <1>      jnz     short pix_op_chr_u ; user font
7420      <1>
7421 0000E65B 80E63F <1>      and     dh, 3Fh ; clear bit 6, [UFONT] option bit
7422 0000E65E 7409 <1>      jz      short pix_op_chr_0
7423      <1>
7424 0000E660 80FE07 <1>      cmp     dh, 7
7425 0000E663 7732 <1>      ja      short pix_op_chr_err
7426      <1>      ; invalid (undefined) option
7427 0000E665 88F4 <1>      mov     ah, dh
7428 0000E667 D0EC <1>      shr     ah, 1
7429      <1>      ; ah = 0 to 3, scale
7430      <1>      ; jmp     short pix_op_chr_font_pixels
7431      <1>
7432      <1> pix_op_chr_font_pixels:
7433      <1>      ; 05/02/2021
7434      <1>      ; write scaled font to buffer
7435      <1>
7436      <1>      ; DL = ASCII code of character
7437      <1>      ; AH = scale
7438      <1>      ; EDI = buffer address (kernel)
7439      <1>
7440      <1> pix_op_chr_0:
7441 0000E669 88D3 <1>      mov     bl, dl ; 15/02/2021
7442 0000E66B 31C9 <1>      xor     ecx, ecx
7443 0000E66D B610 <1>      mov     dh, 16
7444 0000E66F F605[789D0100]01 <1>      test    byte [v_ops], 1 ; 8x8 font ?
7445 0000E676 7428 <1>      jz      short pix_op_chr_2 ; 8x16 font
7446 0000E678 B608 <1>      mov     dh, 8
7447 0000E67A C1E303 <1>      shl     ebx, 3 ; * 8
7448 0000E67D F605[789D0100]40 <1>      test    byte [v_ops], 40h ; [ufont] option
7449 0000E684 7412 <1>      jz      short pix_op_chr_1 ; no
7450      <1>      ; test 8x8 user font is ready flag
7451 0000E686 F605[6E9D0100]01 <1>      test    byte [ufont], 1
7452 0000E68D 7409 <1>      jz      short pix_op_chr_1 ; no
7453 0000E68F 81C300500900 <1>      add     ebx, VGA_FONT8USER
7454 0000E695 EB2C <1>      jmp     short pix_op_chr_fpos_0
7455      <1> pix_op_chr_err:
7456 0000E697 C3 <1>      retn
7457      <1> pix_op_chr_1:
7458 0000E698 81C3[304D0100] <1>      add     ebx, vga_font8 ; system font (8x8)
7459 0000E69E EB23 <1>      jmp     short pix_op_chr_fpos_0
7460      <1> pix_op_chr_2:
7461 0000E6A0 C1E304 <1>      shl     ebx, 4 ; * 16
7462 0000E6A3 F605[789D0100]40 <1>      test    byte [v_ops], 40h ; [ufont] option
7463 0000E6AA 7411 <1>      jz      short pix_op_chr_3 ; no
7464      <1>      ; test 8x16 user font is ready flag
7465 0000E6AC F605[6E9D0100]02 <1>      test    byte [ufont], 2
7466 0000E6B3 7408 <1>      jz      short pix_op_chr_3 ; no
7467 0000E6B5 81C300400900 <1>      add     ebx, VGA_FONT16USER
7468 0000E6BB EB06 <1>      jmp     short pix_op_chr_fpos_0
7469      <1> pix_op_chr_3:
7470 0000E6BD 81C3[30630100] <1>      add     ebx, vga_font16 ; system font (8x16)
7471      <1> pix_op_chr_fpos_0:
7472 0000E6C3 20E4 <1>      and     ah, ah
7473 0000E6C5 754A <1>      jnz     short pix_op_chr_fpos_1 ; scale > 1
7474      <1>      ; no scale (scale = 1)
7475 0000E6C7 89DE <1>      mov     esi, ebx ; 15/02/2021
7476 0000E6C9 88F1 <1>      mov     cl, dh ; rows/height (16 or 8)
7477 0000E6CB B608 <1>      mov     dh, 8 ; columns/width
7478 0000E6CD E9CB000000 <1>      jmp     pix_op_chr_f2p
7479      <1> pix_op_chr_u:
7480      <1>      ; write user defined font
7481 0000E6D2 80FE80 <1>      cmp     dh, 80h
7482 0000E6D5 75C0 <1>      jne     short pix_op_chr_err
7483 0000E6D7 80FA07 <1>      cmp     dl, 7
7484 0000E6DA 77BB <1>      ja      short pix_op_chr_err
7485      <1>
7486      <1>      ; 16/02/2021
7487 0000E6DC 89FE <1>      mov     esi, edi ; user's font buffer
7488      <1>
7489      <1>      ; xor     eax, eax
7490      <1>      ; eax = 0 ; 15/02/2021
7491 0000E6DE 88D4 <1>      mov     ah, dl
7492 0000E6E0 D0EC <1>      shr     ah, 1
7493 0000E6E2 FEC4 <1>      inc     ah
7494      <1>      ; ah = 1 to 4
7495 0000E6E4 88E0 <1>      mov     al, ah
7496 0000E6E6 C0E003 <1>      shl     al, 3 ; * 8
7497      <1>      ; al = 8,16,24,32
7498 0000E6E9 88C3 <1>      mov     bl, al
7499 0000E6EB 88C7 <1>      mov     bh, al
7500 0000E6ED F6E4 <1>      mul     ah
7501      <1>      ; ax = 8,32,72,128 bytes
7502 0000E6EF F6C201 <1>      test    dl, 1
7503 0000E6F2 7404 <1>      jz      short pix_op_chr_u_0
7504      <1>      ; shl     ax, 1 ; *2
7505      <1>      ; 23/07/2022
7506 0000E6F4 D1E0 <1>      shl     eax, 1
7507      <1>      ; ax = 16,32,144,256 bytes
7508 0000E6F6 D0E7 <1>      shl     bh, 1
7509      <1> pix_op_chr_u_0:
7510      <1>      ; eax = byte count
7511 0000E6F8 89C1 <1>      mov     ecx, eax
7512 0000E6FA BF00760900 <1>      mov     edi, VBE3SAVERESTOREBLOCK
7513      <1>      ; esi = user buffer
7514 0000E6FF E851240000 <1>      call    transfer_from_user_buffer
7515 0000E704 7291 <1>      jc      short pix_op_chr_err
7516      <1>
7517 0000E706 88F9 <1>      mov     cl, bh ; rows/height
7518 0000E708 88DE <1>      mov     dh, bl ; columns (width)
7519 0000E70A 89FE <1>      mov     esi, edi ; VBE3SAVERESTOREBLOCK
7520 0000E70C E98C000000 <1>      jmp     pix_op_chr_f2p
7521      <1>
7522      <1> pix_op_chr_fpos_1:
7523      <1>      ; 18/02/2021
7524      <1>      ; scale > 1
7525 0000E711 88F5 <1>      mov     ch, dh ; 16 or 8
7526 0000E713 BF00760900 <1>      mov     edi, VBE3SAVERESTOREBLOCK
7527 0000E718 89FE <1>      mov     esi, edi

```

```

7528 0000E71A FECC      <1>      dec      ah
7529 0000E71C 7522      <1>      jnz      short pix_op_chr_fpos_5 ; scale > 2
7530                      <1>      ; scale = 2
7531                      <1>      pix_op_chr_fpos_2:
7532 0000E71E B108      <1>      mov      cl, 8
7533 0000E720 8A13      <1>      mov      dl, [ebx]
7534                      <1>      pix_op_chr_fpos_3:
7535                      <1>      ;shl      ax, 2
7536                      <1>      ; 23/07/2022
7537 0000E722 C1E002    <1>      shl      eax, 2
7538 0000E725 D0E2      <1>      shl      dl, 1
7539 0000E727 7302      <1>      jnc      short pix_op_chr_fpos_4
7540 0000E729 0C03      <1>      or       al, 3
7541                      <1>      pix_op_chr_fpos_4:
7542 0000E72B FEC9      <1>      dec      cl
7543 0000E72D 75F3      <1>      jnz      short pix_op_chr_fpos_3
7544 0000E72F 66AB      <1>      stosw
7545                      <1>      ; 18/02/2021
7546 0000E731 66AB      <1>      stosw
7547 0000E733 43          <1>      inc      ebx
7548 0000E734 FEC9      <1>      dec      ch
7549 0000E736 75E6      <1>      jnz      short pix_op_chr_fpos_2
7550                      <1>      ; scale = 2
7551 0000E738 88F1      <1>      mov      cl, dh ; 16 or 8 (height/rows)
7552 0000E73A D0E1      <1>      shl      cl, 1 ; 32 or 16 rows
7553 0000E73C B610      <1>      mov      dh, 16 ; columns (width)
7554 0000E73E EB5D      <1>      jmp      short pix_op_chr_f2p
7555                      <1>      pix_op_chr_fpos_5:
7556 0000E740 FECC      <1>      dec      ah
7557 0000E742 7538      <1>      jnz      short pix_op_chr_fpos_9 ; scale = 4
7558                      <1>      ; scale = 3
7559                      <1>      pix_op_chr_fpos_6:
7560 0000E744 B108      <1>      mov      cl, 8
7561 0000E746 8A13      <1>      mov      dl, [ebx]
7562                      <1>      pix_op_chr_fpos_7:
7563 0000E748 C1E003    <1>      shl      eax, 3
7564 0000E74B D0E2      <1>      shl      dl, 1 ; 18/02/2021
7565 0000E74D 7302      <1>      jnc      short pix_op_chr_fpos_8
7566 0000E74F 0C07      <1>      or       al, 7
7567                      <1>      pix_op_chr_fpos_8:
7568 0000E751 FEC9      <1>      dec      cl
7569 0000E753 75F3      <1>      jnz      short pix_op_chr_fpos_7
7570 0000E755 66AB      <1>      stosw
7571                      <1>      ; 18/02/2021
7572 0000E757 C1C810    <1>      ror      eax, 16
7573 0000E75A AA          <1>      stosb
7574 0000E75B C1C010    <1>      rol      eax, 16
7575 0000E75E 66AB      <1>      stosw
7576 0000E760 C1C810    <1>      ror      eax, 16
7577 0000E763 AA          <1>      stosb
7578 0000E764 C1C010    <1>      rol      eax, 16
7579 0000E767 66AB      <1>      stosw
7580 0000E769 C1E810    <1>      shr      eax, 16 ; 27/02/2021
7581 0000E76C AA          <1>      stosb
7582 0000E76D 43          <1>      inc      ebx
7583                      <1>      ; 18/02/2021
7584 0000E76E FEC9      <1>      dec      ch
7585 0000E770 75D2      <1>      jnz      short pix_op_chr_fpos_6
7586                      <1>      ; scale = 3
7587 0000E772 88F1      <1>      mov      cl, dh ; 16 or 8 (height/rows)
7588 0000E774 D0E1      <1>      shl      cl, 1
7589 0000E776 00F1      <1>      add      cl, dh ; 48 or 24 rows
7590 0000E778 B618      <1>      mov      dh, 24 ; columns (width)
7591 0000E77A EB21      <1>      jmp      short pix_op_chr_f2p
7592                      <1>
7593                      <1>      pix_op_chr_fpos_9:
7594                      <1>      ; scale = 4
7595 0000E77C B108      <1>      mov      cl, 8
7596 0000E77E 8A13      <1>      mov      dl, [ebx]
7597                      <1>      pix_op_chr_fpos_10:
7598                      <1>      ; 18/02/2021
7599 0000E780 C1E004    <1>      shl      eax, 4
7600 0000E783 D0E2      <1>      shl      dl, 1 ; 18/02/2021
7601 0000E785 7302      <1>      jnc      short pix_op_chr_fpos_11
7602 0000E787 0C0F      <1>      or       al, 0Fh ; or al, 15
7603                      <1>      pix_op_chr_fpos_11:
7604 0000E789 FEC9      <1>      dec      cl
7605 0000E78B 75F3      <1>      jnz      short pix_op_chr_fpos_10
7606 0000E78D AB          <1>      stosd
7607                      <1>      ; 18/02/2021
7608 0000E78E AB          <1>      stosd
7609 0000E78F AB          <1>      stosd
7610 0000E790 AB          <1>      stosd
7611 0000E791 43          <1>      inc      ebx
7612 0000E792 FEC9      <1>      dec      ch
7613 0000E794 75E6      <1>      jnz      short pix_op_chr_fpos_9
7614                      <1>      ; scale = 4
7615 0000E796 88F1      <1>      mov      cl, dh ; 16 or 8 (height/rows)
7616 0000E798 C0E102    <1>      shl      cl, 2 ; 64 or 32 rows
7617 0000E79B B620      <1>      mov      dh, 32 ; columns (width)
7618                      <1>      ; jmp      short pix_op_chr_f2p
7619                      <1>
7620                      <1>      pix_op_chr_f2p:
7621                      <1>      ; write font pixels
7622 0000E79D 8B3D[829D0100] <1>      mov      edi, [v_str]
7623                      <1>      ; 15/02/2021
7624                      <1>      pix_op_chr_f2p_next:
7625 0000E7A3 80FE08    <1>      cmp      dh, 8
7626 0000E7A6 7706      <1>      ja       short pix_op_chr_f2p_24
7627                      <1>      pix_op_chr_f2p_8:
7628 0000E7A8 AC          <1>      lodsb
7629 0000E7A9 C1E018    <1>      shl      eax, 24 ; 15/02/2021
7630 0000E7AC EB16      <1>      jmp      short pix_op_chr_f2p_0
7631                      <1>      pix_op_chr_f2p_24:
7632 0000E7AE 80FE18    <1>      cmp      dh, 24
7633 0000E7B1 7710      <1>      ja       short pix_op_chr_f2p_32
7634 0000E7B3 7207      <1>      jb       short pix_op_chr_f2p_16
7635                      <1>      ; 27/02/2021
7636                      <1>      ; mov      eax, [esi]
7637 0000E7B5 AD          <1>      lodsd
7638 0000E7B6 C1E008    <1>      shl      eax, 8
7639                      <1>      ; add      esi, 3
7640 0000E7B9 4E          <1>      dec      esi
7641 0000E7BA EB08      <1>      jmp      short pix_op_chr_f2p_0
7642                      <1>      pix_op_chr_f2p_16:
7643 0000E7BC 66AD      <1>      lodsw
7644 0000E7BE C1E010    <1>      shl      eax, 16 ; 15/02/2021
7645 0000E7C1 EB01      <1>      jmp      short pix_op_chr_f2p_0
7646                      <1>      pix_op_chr_f2p_32:
7647 0000E7C3 AD          <1>      lodsd
7648                      <1>      pix_op_chr_f2p_0:
7649                      <1>      ; EAX = font row (8,16,24,32 pixels)
7650                      <1>      ; (bits are shifted to left)
7651                      <1>      ; CL = rows

```

```

7652      ; DH = bits per row (8,16,24,32)
7653 0000E7C4 8B1D[929D0100]    <1>    mov     ebx, [buffer8] ; start column
7654 0000E7CA 57                <1>    push    edi ; *
7655 0000E7CB 52                <1>    push    edx ; **
7656      pix_op_chr_f2p_1:
7657 0000E7CC E82E000000         <1>    call    pix_op_chr_w_pixel
7658      pix_op_chr_f2p_2:
7659 0000E7D1 663B1D[769D0100]    <1>    cmp     bx, [v_width] ; current column
7660 0000E7D8 7304                <1>    jnb     short pix_op_chr_f2p_3
7661 0000E7DA FECE                <1>    dec     dh
7662 0000E7DC 75EE                <1>    jnz     short pix_op_chr_f2p_1 ; next bit
7663      pix_op_chr_f2p_3:
7664      ;mov     ebx, [buffer8]
7665 0000E7DE 5A                <1>    pop     edx ; **
7666 0000E7DF 58                <1>    pop     eax ; *
7667 0000E7E0 3B3D[869D0100]    <1>    cmp     edi, [v_end]
7668 0000E7E6 7316                <1>    jnb     short pix_op_chr_f2p_4
7669 0000E7E8 FEC9                <1>    dec     cl
7670 0000E7EA 7412                <1>    jz      short pix_op_chr_f2p_4
7671      ; 27/02/2021
7672 0000E7EC 89C7                <1>    mov     edi, eax
7673 0000E7EE 0FB705[769D0100]    <1>    movzx   eax, word [v_width]
7674 0000E7F5 E85BF0FFFF         <1>    call    pixels_to_byte_count
7675 0000E7FA 01C7                <1>    add     edi, eax ; next position
7676 0000E7FC EBA5                <1>    jmp     short pix_op_chr_f2p_next
7677      pix_op_chr_f2p_4:
7678 0000E7FE C3                <1>    retn
7679      <1>
7680      pix_op_chr_w_pixel:
7681      ; 15/02/2021
7682 0000E7FF 89C5                <1>    mov     ebp, eax
7683 0000E801 A1[8A9D0100]        <1>    mov     eax, [maskcolor]
7684 0000E806 803D[799D0100]08    <1>    cmp     byte [v_bpp], 8 ; 8bpp
7685 0000E80D 7711                <1>    ja      short pix_op_chr_wp_2
7686      ; 256 colors (1 byte per pixel)
7687 0000E80F D1E5                <1>    shl     ebp, 1
7688 0000E811 7302                <1>    jnc     short pix_op_chr_wp_0
7689 0000E813 8807                <1>    mov     [edi], al
7690      pix_op_chr_wp_0:
7691 0000E815 47                <1>    inc     edi
7692 0000E816 FF05[548E0100]      <1>    inc     dword [u.r0] ; +1
7693      pix_op_chr_wp_1:
7694 0000E81C 43                <1>    inc     ebx
7695 0000E81D 89E8                <1>    mov     eax, ebp
7696 0000E81F C3                <1>    retn
7697      pix_op_chr_wp_2:
7698 0000E820 803D[799D0100]18    <1>    cmp     byte [v_bpp], 24 ; 24bpp
7699 0000E827 772E                <1>    ja      short pix_op_chr_wp_6 ; 32bpp
7700 0000E829 721C                <1>    jb      short pix_op_chr_wp_4 ; 16bpp
7701      ; 24 bit true colors
7702      ; * 3
7703 0000E82B D1E5                <1>    shl     ebp, 1
7704 0000E82D 7309                <1>    jnc     short pix_op_chr_wp_3
7705 0000E82F 668907              <1>    mov     [edi], ax
7706 0000E832 C1E810              <1>    shr     eax, 16
7707 0000E835 884702              <1>    mov     [edi+2], al
7708      pix_op_chr_wp_3:
7709 0000E838 B803000000          <1>    mov     eax, 3 ; 27/02/2021
7710 0000E83D 01C7                <1>    add     edi, eax ; add edi, 3
7711 0000E83F 0105[548E0100]      <1>    add     [u.r0], eax ; +3
7712      <1>
7713 0000E845 EBD5                <1>    jmp     short pix_op_chr_wp_1
7714      <1>
7715      pix_op_chr_wp_4:
7716      ; 16 bit (65536) colors
7717 0000E847 D1E5                <1>    shl     ebp, 1
7718 0000E849 7303                <1>    jnc     short pix_op_chr_wp_5
7719 0000E84B 668907              <1>    mov     [edi], ax
7720      pix_op_chr_wp_5:
7721 0000E84E 47                <1>    inc     edi
7722 0000E84F FF05[548E0100]      <1>    inc     dword [u.r0] ; +1
7723 0000E855 EBBE                <1>    jmp     short pix_op_chr_wp_0
7724      <1>
7725      pix_op_chr_wp_6:
7726      ; 32 bit true colors
7727 0000E857 D1E5                <1>    shl     ebp, 1
7728 0000E859 7302                <1>    jnc     short pix_op_chr_wp_7
7729 0000E85B 8907                <1>    mov     [edi], eax
7730      pix_op_chr_wp_7:
7731 0000E85D 31C0                <1>    xor     eax, eax
7732 0000E85F B004                <1>    mov     al, 4
7733 0000E861 01C7                <1>    add     edi, eax ; add edi, 4
7734 0000E863 0105[548E0100]      <1>    add     [u.r0], eax ; +4
7735 0000E869 EBB1                <1>    jmp     short pix_op_chr_wp_1
7736      <1>
7737      m_pix_op_cpy:
7738      ; 26/02/2021
7739      ; 06/02/2021
7740      ; MASKED COPY PIXELS (full screen)
7741      ;
7742      ; jump from pix_op_cpy
7743      ;
7744      ; INPUT:
7745      ;   ecx = transfer count (bytes)
7746      ;   edi = [v_mem] = start address of LFB
7747      ;   esi = user's buffer address (virtual)
7748      ;
7749      ; OUTPUT:
7750      ;   [u.r0] will be > 0 if succesful
7751      <1>
7752      ; Full screen masked copy
7753      <1>
7754      ; Modified regs: eax, ebx, edx, esi, edi, ecx
7755      <1>
7756      m_pix_op_cpy_0:
7757      ;push     ebx ; *** ; 26/02/2021
7758 0000E86B 57                <1>    push    edi ; **
7759 0000E86C 51                <1>    push    ecx ; *
7760 0000E86D 81F9F8070000         <1>    cmp     ecx, 2040 ; (3*680) ; 26/02/2021
7761 0000E873 7605                <1>    jna     short m_pix_op_cpy_1
7762 0000E875 B9F8070000         <1>    mov     ecx, 2040
7763      m_pix_op_cpy_1:
7764 0000E87A BF00760900          <1>    mov     edi, VBE3SAVERESTOREBLOCK ; temporary buff
7765 0000E87F E8D1220000          <1>    call    transfer_from_user_buffer
7766 0000E884 726C                <1>    jc      short m_pix_op_cpy_3
7767 0000E886 01CE                <1>    add     esi, ecx
7768 0000E888 89F5                <1>    mov     ebp, esi ; save user's buffer address
7769 0000E88A 89FE                <1>    mov     esi, edi
7770 0000E88C 89CB                <1>    mov     ebx, ecx
7771 0000E88E 59                <1>    pop     ecx ; *
7772 0000E88F 29D9                <1>    sub     ecx, ebx
7773 0000E891 5F                <1>    pop     edi ; **
7774 0000E892 31D2                <1>    xor     edx, edx ; 26/02/2021
7775 0000E894 803D[799D0100]08    <1>    cmp     byte [v_bpp], 8

```

```

7776 0000E89B 7435      <1>      je      short m_pix_op_cpy_1_8
7777 0000E89D 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24
7778 0000E8A4 776D      <1>      ja      short m_pix_op_cpy_1_32
7779 0000E8A6 724D      <1>      jb      short m_pix_op_cpy_1_16
7780      <1> m_pix_op_cpy_1_24:
7781      <1>      ; 24 bit masked copy
7782      <1>      ;mov     edx, 3
7783 0000E8A8 B203      <1>      mov     dl, 3 ; 26/02/2021
7784      <1> m_pix_op_cpy_1_24_0:
7785 0000E8AA 66AD      <1>      lodsw
7786 0000E8AC C1E010      <1>      shl     eax, 16
7787 0000E8AF AC          <1>      lodsb
7788 0000E8B0 C1C010      <1>      rol     eax, 16
7789 0000E8B3 3B05[8A9D0100] <1>      cmp     eax, [maskcolor]
7790 0000E8B9 740F      <1>      je      short m_pix_op_cpy_1_24_1 ; exclude
7791 0000E8BB 668907      <1>      mov     [edi], ax
7792 0000E8BE C1E810      <1>      shr     eax, 16
7793 0000E8C1 884702      <1>      mov     [edi+2], al
7794 0000E8C4 0115[548E0100] <1>      add     [u.r0], edx ; +3
7795      <1> m_pix_op_cpy_1_24_1:
7796 0000E8CA 01D7      <1>      add     edi, edx ; +3
7797 0000E8CC 29D3      <1>      sub     ebx, edx ; sub ebx, 3
7798 0000E8CE 77DA      <1>      ja      short m_pix_op_cpy_1_24_0
7799 0000E8D0 EB15      <1>      jmp     short m_pix_op_cpy_2
7800      <1>
7801      <1> m_pix_op_cpy_1_8:
7802      <1>      ; 8 bit masked copy
7803 0000E8D2 AC          <1>      lodsb
7804 0000E8D3 3A05[8A9D0100] <1>      cmp     al, [maskcolor]
7805 0000E8D9 7408      <1>      je      short m_pix_op_cpy_1_8_1 ; exclude
7806 0000E8DB 8807      <1>      mov     [edi], al
7807 0000E8DD FF05[548E0100] <1>      inc     dword [u.r0] ; +1
7808      <1> m_pix_op_cpy_1_8_1:
7809 0000E8E3 47          <1>      inc     edi ; +1
7810 0000E8E4 4B          <1>      dec     ebx
7811 0000E8E5 75EB      <1>      jnz     short m_pix_op_cpy_1_8
7812      <1> m_pix_op_cpy_2:
7813 0000E8E7 89EE      <1>      mov     esi, ebp ; restore user's buffer addr
7814 0000E8E9 09C9      <1>      or      ecx, ecx
7815      <1>      ; 26/02/2021
7816 0000E8EB 7407      <1>      jz      short m_pix_of_cpy_4
7817 0000E8ED E979FFFFFF <1>      jmp     m_pix_op_cpy_0
7818      <1> m_pix_op_cpy_3:
7819 0000E8F2 59          <1>      pop     ecx ; *
7820 0000E8F3 5F          <1>      pop     edi ; **
7821      <1> m_pix_of_cpy_4:
7822      <1>      ;pop     ebx ; *** ; 26/02/2021
7823 0000E8F4 C3          <1>      retn
7824      <1>
7825      <1> m_pix_op_cpy_1_16:
7826      <1>      ; 16 bit masked copy
7827      <1>      ;mov     edx, 2
7828 0000E8F5 B202      <1>      mov     dl, 2 ; 26/02/2021
7829      <1> m_pix_op_cpy_1_16_0:
7830 0000E8F7 66AD      <1>      lodsw
7831 0000E8F9 663B05[8A9D0100] <1>      cmp     ax, [maskcolor]
7832 0000E900 7409      <1>      je      short m_pix_op_cpy_1_16_1 ; exclude
7833 0000E902 668907      <1>      mov     [edi], ax
7834 0000E905 0115[548E0100] <1>      add     [u.r0], edx ; +2
7835      <1> m_pix_op_cpy_1_16_1:
7836 0000E90B 01D7      <1>      add     edi, edx ; +2
7837 0000E90D 29D3      <1>      sub     ebx, edx ; sub ebx, 2
7838 0000E90F 77E6      <1>      ja      short m_pix_op_cpy_1_16_0
7839 0000E911 EBD4      <1>      jmp     short m_pix_op_cpy_2
7840      <1>
7841      <1> m_pix_op_cpy_1_32:
7842      <1>      ; 32 bit masked copy
7843      <1>      ;mov     edx, 4
7844 0000E913 B204      <1>      mov     dl, 4 ; 26/02/2021
7845      <1> m_pix_op_cpy_1_32_0:
7846 0000E915 AD          <1>      lodsd
7847 0000E916 3B05[8A9D0100] <1>      cmp     eax, [maskcolor]
7848 0000E91C 7408      <1>      je      short m_pix_op_cpy_1_32_1 ; exclude
7849 0000E91E 8907      <1>      mov     [edi], eax
7850 0000E920 0115[548E0100] <1>      add     [u.r0], edx ; +4
7851      <1> m_pix_op_cpy_1_32_1:
7852 0000E926 01D7      <1>      add     edi, edx ; +4
7853 0000E928 29D3      <1>      sub     ebx, edx ; sub ebx, 4
7854 0000E92A 77E9      <1>      ja      short m_pix_op_cpy_1_32_0
7855 0000E92C EBB9      <1>      jmp     short m_pix_op_cpy_2
7856      <1>
7857      <1> m_pix_op_cpy_w:
7858      <1>      ; 26/02/2021
7859      <1>      ; 06/02/2021
7860      <1>      ; MASKED COPY PIXELS (window)
7861      <1>      ;
7862      <1>      ; jump from pix_op_cpy_w
7863      <1>      ;
7864      <1>      ; INPUT:
7865      <1>      ; ecx = bytes per row (to be applied)
7866      <1>      ; edx = screen width in bytes
7867      <1>      ; ebx = row count
7868      <1>      ; esi = user's buffer address
7869      <1>      ; [v_str] = window start address
7870      <1>      ;
7871      <1>      ; OUTPUT:
7872      <1>      ; [u.r0] will be > 0 if succesful
7873      <1>      ;
7874      <1>      ; window masked copy
7875      <1>
7876      <1> m_pix_op_cpy_w_0:
7877 0000E92E 8B3D[829D0100] <1>      mov     edi, [v_str]
7878      <1> m_pix_op_cpy_w_1:
7879 0000E934 57          <1>      push     edi
7880 0000E935 56          <1>      push     esi
7881 0000E936 53          <1>      push     ebx
7882 0000E937 52          <1>      push     edx
7883 0000E938 51          <1>      push     ecx
7884 0000E939 E82DFFFFFF <1>      call    m_pix_op_cpy ; 26/02/2021
7885 0000E93E 59          <1>      pop      ecx
7886 0000E93F 5A          <1>      pop      edx
7887 0000E940 5B          <1>      pop      ebx
7888 0000E941 5E          <1>      pop      esi
7889 0000E942 5F          <1>      pop      edi
7890 0000E943 7209      <1>      jc      short m_pix_op_cpy_w_2
7891 0000E945 4B          <1>      dec     ebx
7892 0000E946 7406      <1>      jz      short m_pix_op_cpy_w_2 ; ok.
7893      <1>      ; next row
7894 0000E948 01CE      <1>      add     esi, ecx ; next row in user's buffer
7895 0000E94A 01D7      <1>      add     edi, edx ; next row of window (system)
7896 0000E94C EBE6      <1>      jmp     short m_pix_op_cpy_w_1
7897      <1> m_pix_op_cpy_w_2:
7898 0000E94E C3          <1>      retn
7899      <1>

```

```

7900      <1> m_pix_op_new:
7901      <1> ; 06/02/2021
7902      <1> ; CHANGE COLOR (MASKED, full screen)
7903      <1> ;
7904      <1> ; jump from pix_op_new
7905      <1> ;
7906      <1> ; INPUT:
7907      <1> ;     eax = color (AL, AX, EAX)
7908      <1> ;     ecx = [v_siz] ; display page pixel count
7909      <1> ;     esi = edi = [v_mem] ; LFB start address
7910      <1> ;
7911      <1> ;     [maskcolor] = mask color (to be excluded)
7912      <1> ;
7913      <1> ; OUTPUT:
7914      <1> ;     [u.r0] will be > 0 if succesful
7915      <1> ;
7916      <1> ; Full screen
7917      <1> m_pix_op_new_0:
7918      <1> cmp     byte [v_bpp], 8 ; 8bpp
7919      <1> ja      short m_pix_op_new_1
7920      <1> ; 256 colors (8bpp)
7921      <1> jmp     short m_pix_op_new_8
7922      <1> m_pix_op_new_8:
7923      <1> ; 8 bit colors (256 colors)
7924      <1> mov     dl, al ; new color
7925      <1> m_pix_op_new_8_0:
7926      <1> lodsb
7927      <1> cmp     al, [maskcolor]
7928      <1> je      short m_pix_op_new_8_1 ; exclude
7929      <1> mov     [edi], dl
7930      <1> inc     dword [u.r0]
7931      <1> m_pix_op_new_8_1:
7932      <1> inc     edi
7933      <1> loop    m_pix_op_new_8_0
7934      <1> retn
7935      <1> m_pix_op_new_1:
7936      <1> cmp     byte [v_bpp], 24 ; 24bpp
7937      <1> ja      short m_pix_op_new_3 ; 32bpp
7938      <1> jnb     short m_pix_op_new_2 ; 16bpp
7939      <1> ; 24 bit true colors
7940      <1> jmp     short m_pix_op_new_24
7941      <1> m_pix_op_new_24:
7942      <1> ; 24 bit true colors
7943      <1> mov     edx, eax ; new color
7944      <1> m_pix_op_new_24_0:
7945      <1> lodsw
7946      <1> shl     eax, 16
7947      <1> lodsb
7948      <1> rol     eax, 16
7949      <1> cmp     eax, [maskcolor]
7950      <1> je      short m_pix_op_new_24_1 ; exclude
7951      <1> mov     [edi], dx
7952      <1> ror     edx, 16
7953      <1> mov     [edi+2], dl
7954      <1> rol     edx, 16
7955      <1> add     dword [u.r0], 3
7956      <1> m_pix_op_new_24_1:
7957      <1> add     edi, 3
7958      <1> loop    m_pix_op_new_24_0
7959      <1> retn
7960      <1> ; 65536 colors (16bpp)
7961      <1> m_pix_op_new_2:
7962      <1> jmp     short m_pix_op_new_16
7963      <1> m_pix_op_new_16:
7964      <1> ; 16 bit colors (65536 colors)
7965      <1> mov     edx, eax ; new color
7966      <1> m_pix_op_new_16_0:
7967      <1> lodsw
7968      <1> cmp     ax, [maskcolor]
7969      <1> je      short m_pix_op_new_16_1 ; exclude
7970      <1> mov     [edi], dx
7971      <1> add     dword [u.r0], 2
7972      <1> m_pix_op_new_16_1:
7973      <1> add     edi, 2
7974      <1> loop    m_pix_op_new_16_0
7975      <1> retn
7976      <1> m_pix_op_new_3:
7977      <1> ; 32 bit true colors
7978      <1> jmp     short m_pix_op_new_32
7979      <1> m_pix_op_new_32:
7980      <1> ; 32 bit true colors
7981      <1> mov     edx, eax ; new color
7982      <1> m_pix_op_new_32_0:
7983      <1> lodsd
7984      <1> cmp     eax, [maskcolor]
7985      <1> je      short m_pix_op_new_32_1 ; exclude
7986      <1> mov     [edi], edx
7987      <1> add     dword [u.r0], 4
7988      <1> m_pix_op_new_32_1:
7989      <1> add     edi, 4
7990      <1> loop    m_pix_op_new_32_0
7991      <1> retn
7992      <1>
7993      <1> m_pix_op_new_w:
7994      <1> ; 06/02/2021
7995      <1> ; CHANGE COLOR (MASKED, window)
7996      <1> ;
7997      <1> ; jump from pix_op_new_w
7998      <1> ;
7999      <1> ; INPUT:
8000      <1> ;     ecx = bytes per row (to be applied)
8001      <1> ;     edx = screen width in bytes
8002      <1> ;     ebx = row count
8003      <1> ;     eax = color
8004      <1> ;
8005      <1> ;     [maskcolor] = mask color (to be excluded)
8006      <1> ;
8007      <1> ; OUTPUT:
8008      <1> ;     [u.r0] will be > 0 if succesful
8009      <1> ;
8010      <1> ; window
8011      <1> ; mov     edi, [v_str] ; LFB start address
8012      <1> ; mov     esi, edi
8013      <1> ;
8014      <1> cmp     byte [v_bpp], 8 ; 8bpp
8015      <1> ja      short m_pix_op_new_w_1
8016      <1> ;
8017      <1> ; 256 colors (8bpp)
8018      <1> mov     ebp, m_pix_op_new_8
8019      <1> jmp     short m_pix_op_new_w_x
8020      <1> ;
8021      <1> m_pix_op_new_w_1:
8022      <1> cmp     byte [v_bpp], 24 ; 24bpp
8023      <1> ja      short m_pix_op_new_w_3 ; 32bpp

```



```

8024 0000E9F6 7207      <1>      jb      short m_pix_op_new_w_2 ; 16bpp
8025                                <1>
8026                                <1>      ; 24 bit true colors
8027 0000E9F8 BD[7AE90000] <1>      mov     ebp, m_pix_op_new_24
8028 0000E9FD EB0C      <1>      jmp     short m_pix_op_new_w_x
8029                                <1>
8030                                <1>      ; 65536 colors (16bpp)
8031                                <1> m_pix_op_new_w_2:
8032 0000E9FF BD[A6E90000] <1>      mov     ebp, m_pix_op_new_16
8033 0000EA04 EB05      <1>      jmp     short m_pix_op_new_w_x
8034                                <1>
8035                                <1>      ; 32 bit true colors
8036                                <1> m_pix_op_new_w_3:
8037 0000EA06 BD[C3E90000] <1>      mov     ebp, m_pix_op_new_32
8038                                <1>      ; jmp     short m_pix_op_new_w_x
8039                                <1>
8040                                <1> m_pix_op_new_w_x:
8041                                <1> m_pix_op_add_w_x:
8042                                <1> m_pix_op_sub_w_x:
8043                                <1> m_pix_op_mix_w_x:
8044                                <1> m_pix_op_and_w_x:
8045                                <1> m_pix_op_orc_w_x:
8046                                <1> m_pix_op_xor_w_x:
8047                                <1> m_pix_op_not_w_x:
8048                                <1> m_pix_op_neg_w_x:
8049                                <1> m_pix_op_inc_w_x:
8050                                <1> m_pix_op_dec_w_x:
8051                                <1>      ; 27/02/2021
8052                                <1>      ; 26/02/2021
8053                                <1>      ; 06/02/2021
8054                                <1>      ; ecx = bytes per row (to be applied)
8055                                <1>      ; edx = windows (screen) width in bytes
8056                                <1>      ; ebx = row count
8057                                <1>      ; eax = color
8058                                <1>      ; ebp = pixel operation subroutine address
8059                                <1>      ; edi = esi = window start address
8060                                <1>
8061 0000EA0B 8B3D[829D0100] <1>      mov     edi, [v_str] ; LFB start address
8062 0000EA11 89FE      <1>      mov     esi, edi
8063                                <1> m_pix_op_w_x_next:
8064 0000EA13 52          <1>      push    edx
8065 0000EA14 51          <1>      push    ecx
8066 0000EA15 56          <1>      push    esi
8067 0000EA16 57          <1>      push    edi
8068 0000EA17 50          <1>      push    eax ; 26/02/2021
8069 0000EA18 8B0D[8E9D0100] <1>      mov     ecx, [pixcount] ; 27/02/2021
8070 0000EA1E FFD5      <1>      call    ebp ; call masked pixel-row operation
8071 0000EA20 58          <1>      pop     eax ; 26/02/2021
8072 0000EA21 5F          <1>      pop     edi
8073 0000EA22 5E          <1>      pop     esi
8074 0000EA23 59          <1>      pop     ecx
8075 0000EA24 5A          <1>      pop     edx
8076 0000EA25 01D6      <1>      add     esi, edx ; next row
8077 0000EA27 01D7      <1>      add     edi, edx ; next row
8078 0000EA29 4B          <1>      dec     ebx
8079 0000EA2A 75E7      <1>      jnz     short m_pix_op_w_x_next
8080 0000EA2C C3          <1>      retn
8081                                <1>
8082                                <1> m_pix_op_add:
8083                                <1>      ; 06/02/2021
8084                                <1>      ; ADD COLOR (MASKED, full screen)
8085                                <1>      ;
8086                                <1>      ; jump from pix_op_add
8087                                <1>      ;
8088                                <1>      ; INPUT:
8089                                <1>      ;     eax = color (AL, AX, EAX)
8090                                <1>      ;     ecx = [v_siz] ; display page pixel count
8091                                <1>      ;     esi = edi = [v_mem] ; LFB start address
8092                                <1>      ;
8093                                <1>      ;     [maskcolor] = mask color (to be excluded)
8094                                <1>      ;
8095                                <1>      ; OUTPUT:
8096                                <1>      ;     [u.r0] will be > 0 if succesful
8097                                <1>      ;
8098                                <1>      ; Full screen
8099                                <1> m_pix_op_add_0:
8100 0000EA2D 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
8101 0000EA34 771C      <1>      ja     short m_pix_op_add_1
8102                                <1>      ; 256 colors (8bpp)
8103                                <1>      ; jmp     short m_pix_op_add_8
8104                                <1> m_pix_op_add_8:
8105                                <1>      ; 8 bit colors (256 colors)
8106 0000EA36 88C2      <1>      mov     dl, al ; new color
8107                                <1> m_pix_op_add_8_0:
8108 0000EA38 AC          <1>      lodsb
8109 0000EA39 3A05[8A9D0100] <1>      cmp     al, [maskcolor]
8110 0000EA3F 740D      <1>      je     short m_pix_op_add_8_1 ; exclude
8111 0000EA41 FF05[548E0100] <1>      inc     dword [u.r0] ; +1
8112 0000EA47 0017      <1>      add     [edi], dl
8113 0000EA49 7303      <1>      jnc     short m_pix_op_add_8_1
8114 0000EA4B C607FF      <1>      mov     byte [edi], 0FFh
8115                                <1> m_pix_op_add_8_1:
8116 0000EA4E 47          <1>      inc     edi
8117 0000EA4F E2E7      <1>      loop    m_pix_op_add_8_0
8118 0000EA51 C3          <1>      retn
8119                                <1> m_pix_op_add_1:
8120 0000EA52 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
8121 0000EA59 775E      <1>      ja     short m_pix_op_add_3 ; 32bpp
8122 0000EA5B 7238      <1>      jb     short m_pix_op_add_2 ; 16bpp
8123                                <1>      ; 24 bit true colors
8124                                <1>      ; jmp     short m_pix_op_add_24
8125                                <1> m_pix_op_add_24:
8126                                <1>      ; 24 bit true colors
8127 0000EA5D 89C2      <1>      mov     edx, eax ; new color
8128 0000EA5F 81CA000000FF <1>      or      edx, 0FF00000h
8129                                <1> m_pix_op_add_24_0:
8130                                <1>      lodsw
8131 0000EA67 C1E010      <1>      shl     eax, 16
8132 0000EA6A AC          <1>      lodsb
8133 0000EA6B C1C010      <1>      rol     eax, 16
8134 0000EA6E 3B05[8A9D0100] <1>      cmp     eax, [maskcolor]
8135 0000EA74 7419      <1>      je     short m_pix_op_add_24_2 ; exclude
8136 0000EA76 8305[548E0100]03 <1>      add     dword [u.r0], 3 ; +3
8137 0000EA7D 01D0      <1>      add     eax, edx
8138 0000EA7F 7305      <1>      jnc     short m_pix_op_add_24_1
8139 0000EA81 B8FFFFFFF00 <1>      mov     eax, 0FFFFFFh
8140                                <1> m_pix_op_add_24_1:
8141 0000EA86 668907      <1>      mov     [edi], ax
8142 0000EA89 C1E810      <1>      shr     eax, 16
8143 0000EA8C 884702      <1>      mov     [edi+2], al
8144                                <1> m_pix_op_add_24_2:
8145 0000EA8F 83C703      <1>      add     edi, 3 ; +3
8146 0000EA92 E2D1      <1>      loop    m_pix_op_add_24_0
8147 0000EA94 C3          <1>      retn

```

```

8148      ; 65536 colors (16bpp)
8149      <1> m_pix_op_add_2:
8150      <1> ; jmp short m_pix_op_add_16
8151      <1> m_pix_op_add_16:
8152      <1> ; 16 bit colors (65536 colors)
8153      0000EA95 89C2      <1> mov     edx, eax ; new color
8154      <1> m_pix_op_add_16_0:
8155      0000EA97 66AD      <1> lodsw
8156      0000EA99 663B05[8A9D0100] <1> cmp     ax, [maskcolor]
8157      0000EAA0 7411      <1> je      short m_pix_op_add_16_1 ; exclude
8158      0000EAA2 8305[548E0100]02 <1> add     dword [u.r0], 2 ; +2
8159      0000EAA9 660117      <1> add     [edi], dx
8160      0000EAA0 7305      <1> jnc     short m_pix_op_add_16_1
8161      0000EAAE 66C707FFFF <1> mov     word [edi], 0FFFFh
8162      <1> m_pix_op_add_16_1:
8163      0000EAB3 83C702      <1> add     edi, 2 ; +2
8164      0000EAB6 E2DF      <1> loop    m_pix_op_add_16_0
8165      0000EAB8 C3      <1> retn
8166      <1> m_pix_op_add_3:
8167      <1> ; 32 bit true colors
8168      <1> ; jmp short m_pix_op_add_32
8169      <1> m_pix_op_add_32:
8170      <1> ; 32 bit true colors
8171      0000EAB9 89C2      <1> mov     edx, eax ; new color
8172      <1> m_pix_op_add_32_0:
8173      0000EABB AD      <1> lodsd
8174      0000EABC 3B05[8A9D0100] <1> cmp     eax, [maskcolor]
8175      0000EAC2 7411      <1> je      short m_pix_op_add_32_1 ; exclude
8176      0000EAC4 8305[548E0100]04 <1> add     dword [u.r0], 4 ; +4
8177      0000EACB 0117      <1> add     [edi], edx
8178      0000EACD 7306      <1> jnc     short m_pix_op_add_32_1
8179      0000EACF C707FFFFFFFF <1> mov     dword [edi], 0FFFFFFFFh
8180      <1> m_pix_op_add_32_1:
8181      0000EAD5 83C704      <1> add     edi, 4 ; +4
8182      0000EAD8 E2E1      <1> loop    m_pix_op_add_32_0
8183      0000EADA C3      <1> retn
8184      <1>
8185      <1> m_pix_op_add_w:
8186      <1> ; 06/02/2021
8187      <1> ; ADD COLOR (MASKED, window)
8188      <1> ;
8189      <1> ; jump from pix_op_add_w
8190      <1> ;
8191      <1> ; INPUT:
8192      <1> ; ecx = bytes per row (to be applied)
8193      <1> ; edx = screen width in bytes
8194      <1> ; ebx = row count
8195      <1> ; eax = color
8196      <1> ;
8197      <1> ; [maskcolor] = mask color (to be excluded)
8198      <1> ;
8199      <1> ; OUTPUT:
8200      <1> ; [u.r0] will be > 0 if succesful
8201      <1> ;
8202      <1> ; window
8203      <1> ; mov edi, [v_str] ; LFB start address
8204      <1> ; mov esi, edi
8205      <1> ;
8206      0000EADB 803D[799D0100]08 <1> cmp     byte [v_bpp], 8 ; 8bpp
8207      0000EAE2 7707      <1> ja      short m_pix_op_add_w_1
8208      <1> ;
8209      <1> ; 256 colors (8bpp)
8210      0000EAE4 BD[36EA0000] <1> mov     ebp, m_pix_op_add_8
8211      0000EAE9 EB1E      <1> jmp     short m_pix_op_add_w_4
8212      <1> ;
8213      <1> m_pix_op_add_w_1:
8214      0000EAEB 803D[799D0100]18 <1> cmp     byte [v_bpp], 24 ; 24bpp
8215      0000EAF2 7710      <1> ja      short m_pix_op_add_w_3 ; 32bpp
8216      0000EAF4 7207      <1> jb      short m_pix_op_add_w_2 ; 16bpp
8217      <1> ;
8218      <1> ; 24 bit true colors
8219      0000EAF6 BD[5DEA0000] <1> mov     ebp, m_pix_op_add_24
8220      0000EAFB EB0C      <1> jmp     short m_pix_op_add_w_4
8221      <1> ;
8222      <1> ; 65536 colors (16bpp)
8223      <1> m_pix_op_add_w_2:
8224      0000EAFD BD[95EA0000] <1> mov     ebp, m_pix_op_add_16
8225      0000EB02 EB05      <1> jmp     short m_pix_op_add_w_4
8226      <1> ;
8227      <1> ; 32 bit true colors
8228      <1> m_pix_op_add_w_3:
8229      0000EB04 BD[B9EA0000] <1> mov     ebp, m_pix_op_add_32
8230      <1> m_pix_op_add_w_4:
8231      0000EB09 E9FDFEFFFF <1> jmp     m_pix_op_add_w_x
8232      <1>
8233      <1> m_pix_op_sub:
8234      <1> ; 02/03/2021
8235      <1> ; 06/02/2021
8236      <1> ; SUBTRACT COLOR (MASKED, full screen)
8237      <1> ;
8238      <1> ; jump from pix_op_sub
8239      <1> ;
8240      <1> ; INPUT:
8241      <1> ; eax = color (AL, AX, EAX)
8242      <1> ; ecx = [v_siz] ; display page pixel count
8243      <1> ; esi = edi = [v_mem] ; LFB start address
8244      <1> ;
8245      <1> ; [maskcolor] = mask color (to be excluded)
8246      <1> ;
8247      <1> ; OUTPUT:
8248      <1> ; [u.r0] will be > 0 if succesful
8249      <1> ;
8250      <1> ; Full screen
8251      <1> m_pix_op_sub_0:
8252      0000EB0E 803D[799D0100]08 <1> cmp     byte [v_bpp], 8 ; 8bpp
8253      0000EB15 771C      <1> ja      short m_pix_op_sub_1
8254      <1> ; 256 colors (8bpp)
8255      <1> ; jmp short m_pix_op_sub_8
8256      <1> m_pix_op_sub_8:
8257      <1> ; 8 bit colors (256 colors)
8258      0000EB17 88C2      <1> mov     dl, al ; new color
8259      <1> m_pix_op_sub_8_0:
8260      0000EB19 AC      <1> lodsb
8261      0000EB1A 3A05[8A9D0100] <1> cmp     al, [maskcolor]
8262      0000EB20 740D      <1> je      short m_pix_op_sub_8_1 ; exclude
8263      0000EB22 FF05[548E0100] <1> inc     dword [u.r0] ; +1
8264      0000EB28 2817      <1> sub     [edi], dl
8265      0000EB2A 7303      <1> jnb     short m_pix_op_sub_8_1
8266      0000EB2C C60700      <1> mov     byte [edi], 0
8267      <1> m_pix_op_sub_8_1:
8268      0000EB2F 47      <1> inc     edi
8269      0000EB30 E2E7      <1> loop    m_pix_op_sub_8_0
8270      0000EB32 C3      <1> retn
8271      <1> m_pix_op_sub_1:

```

```

8272 0000EB33 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
8273 0000EB3A 7755 <1> ja short m_pix_op_sub_3 ; 32bpp
8274 0000EB3C 722F <1> jb short m_pix_op_sub_2 ; 16bpp
8275 <1> ; 24 bit true colors
8276 <1> ; jmp short m_pix_op_sub_24
8277 <1> m_pix_op_sub_24:
8278 <1> ; 24 bit true colors
8279 0000EB3E 89C2 <1> mov edx, eax ; new color
8280 <1> ; 02/03/2021
8281 <1> m_pix_op_sub_24_0:
8282 <1> lodsw
8283 0000EB42 C1E010 <1> shl eax, 16
8284 0000EB45 AC <1> lodsb
8285 0000EB46 C1C010 <1> rol eax, 16
8286 0000EB49 3B05[8A9D0100] <1> cmp eax, [maskcolor]
8287 0000EB4F 7416 <1> je short m_pix_op_sub_24_2 ; exclude
8288 0000EB51 8305[548E0100]03 <1> add dword [u.r0], 3 ; +3
8289 0000EB58 29D0 <1> sub eax, edx
8290 0000EB5A 7302 <1> jnb short m_pix_op_sub_24_1
8291 0000EB5C 31C0 <1> xor eax, eax ; 0
8292 <1> m_pix_op_sub_24_1:
8293 0000EB5E 668907 <1> mov [edi], ax
8294 0000EB61 C1E810 <1> shr eax, 16
8295 0000EB64 884702 <1> mov [edi+2], al
8296 <1> m_pix_op_sub_24_2:
8297 0000EB67 83C703 <1> add edi, 3 ; +3
8298 0000EB6A E2D4 <1> loop m_pix_op_sub_24_0
8299 0000EB6C C3 <1> retn
8300 <1> ; 65536 colors (16bpp)
8301 <1> m_pix_op_sub_2:
8302 <1> ; jmp short m_pix_op_sub_16
8303 <1> m_pix_op_sub_16:
8304 <1> ; 16 bit colors (65536 colors)
8305 0000EB6D 89C2 <1> mov edx, eax ; new color
8306 <1> m_pix_op_sub_16_0:
8307 0000EB6F 66AD <1> lodsw
8308 0000EB71 663B05[8A9D0100] <1> cmp ax, [maskcolor]
8309 0000EB78 7411 <1> je short m_pix_op_sub_16_1 ; exclude
8310 0000EB7A 8305[548E0100]02 <1> add dword [u.r0], 2 ; +2
8311 0000EB81 662917 <1> sub [edi], dx
8312 0000EB84 7305 <1> jnb short m_pix_op_sub_16_1
8313 0000EB86 31C0 <1> xor eax, eax
8314 0000EB88 668907 <1> mov [edi], ax ; 0
8315 <1> m_pix_op_sub_16_1:
8316 0000EB8B 83C702 <1> add edi, 2 ; +2
8317 0000EB8E E2DF <1> loop m_pix_op_sub_16_0
8318 0000EB90 C3 <1> retn
8319 <1> m_pix_op_sub_3:
8320 <1> ; 32 bit true colors
8321 <1> ; jmp short m_pix_op_sub_32
8322 <1> m_pix_op_sub_32:
8323 <1> ; 32 bit true colors
8324 0000EB91 89C2 <1> mov edx, eax ; new color
8325 <1> m_pix_op_sub_32_0:
8326 0000EB93 AD <1> lodsd
8327 0000EB94 3B05[8A9D0100] <1> cmp eax, [maskcolor]
8328 0000EB9A 740F <1> je short m_pix_op_sub_32_1 ; exclude
8329 0000EB9C 8305[548E0100]04 <1> add dword [u.r0], 4 ; +4
8330 0000EBA3 2917 <1> sub [edi], edx
8331 0000EBA5 7304 <1> jnb short m_pix_op_sub_32_1
8332 0000EBA7 31C0 <1> xor eax, eax
8333 0000EBA9 8907 <1> mov [edi], eax ; 0
8334 <1> m_pix_op_sub_32_1:
8335 0000EBAB 83C704 <1> add edi, 4 ; +4
8336 0000EBAE E2E3 <1> loop m_pix_op_sub_32_0
8337 0000EBB0 C3 <1> retn
8338 <1>
8339 <1> m_pix_op_sub_w:
8340 <1> ; 06/02/2021
8341 <1> ; SUBTRACT COLOR (MASKED, window)
8342 <1> ;
8343 <1> ; jump from pix_op_sub_w
8344 <1> ;
8345 <1> ; INPUT:
8346 <1> ; ecx = bytes per row (to be applied)
8347 <1> ; edx = screen width in bytes
8348 <1> ; ebx = row count
8349 <1> ; eax = color
8350 <1> ;
8351 <1> ; [maskcolor] = mask color (to be excluded)
8352 <1> ;
8353 <1> ; OUTPUT:
8354 <1> ; [u.r0] will be > 0 if succesful
8355 <1> ;
8356 <1> ; window
8357 <1> ; mov edi, [v_str] ; LFB start address
8358 <1> ; mov esi, edi
8359 <1>
8360 0000EBB1 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
8361 0000EBB8 7707 <1> ja short m_pix_op_sub_w_1
8362 <1>
8363 <1> ; 256 colors (8bpp)
8364 0000EBBA BD[17EB0000] <1> mov ebp, m_pix_op_sub_8
8365 0000EBBF EB1E <1> jmp short m_pix_op_sub_w_4
8366 <1>
8367 <1> m_pix_op_sub_w_1:
8368 0000EBBC 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
8369 0000EBC8 7710 <1> ja short m_pix_op_sub_w_3 ; 32bpp
8370 0000EBCA 7207 <1> jb short m_pix_op_sub_w_2 ; 16bpp
8371 <1>
8372 <1> ; 24 bit true colors
8373 0000EBCC BD[3EEB0000] <1> mov ebp, m_pix_op_sub_24
8374 0000EBD1 EB0C <1> jmp short m_pix_op_sub_w_4
8375 <1>
8376 <1> ; 65536 colors (16bpp)
8377 <1> m_pix_op_sub_w_2:
8378 0000EBD3 BD[6DEB0000] <1> mov ebp, m_pix_op_sub_16
8379 0000EBD8 EB05 <1> jmp short m_pix_op_sub_w_4
8380 <1>
8381 <1> ; 32 bit true colors
8382 <1> m_pix_op_sub_w_3:
8383 0000EBDA BD[91EB0000] <1> mov ebp, m_pix_op_sub_32
8384 <1> m_pix_op_sub_w_4:
8385 0000EBDF E927FEFFFF <1> jmp m_pix_op_sub_w_x
8386 <1>
8387 <1> m_pix_op_mix:
8388 <1> ; 25/02/2021
8389 <1> ; 06/02/2021
8390 <1> ; MIX COLOR (MASKED, full screen)
8391 <1> ;
8392 <1> ; jump from pix_op_mix
8393 <1> ;
8394 <1> ; INPUT:
8395 <1> ; eax = color (AL, AX, EAX)

```

```

8396      <1>      ; ecx = [v_siz] ; display page pixel count
8397      <1>      ; esi = edi = [v_mem] ; LFB start address
8398      <1>      ;
8399      <1>      ; [maskcolor] = mask color (to be excluded)
8400      <1>      ;
8401      <1>      ; OUTPUT:
8402      <1>      ; [u.r0] will be > 0 if succesful
8403      <1>      ;
8404      <1>      ; Full screen
8405      <1>      m_pix_op_mix_0:
8406      <1>      cmp     byte [v_bpp], 8 ; 8bpp
8407      <1>      ja      short m_pix_op_mix_1
8408      <1>      ; 256 colors (8bpp)
8409      <1>      jmp     short m_pix_op_mix_8
8410      <1>      m_pix_op_mix_8:
8411      <1>      ; 8 bit colors (256 colors)
8412      <1>      mov     dl, al ; new (mixing) color
8413      <1>      m_pix_op_mix_8_0:
8414      <1>      lodsb
8415      <1>      cmp     al, [maskcolor]
8416      <1>      je      short m_pix_op_mix_8_1 ; exclude
8417      <1>      add     al, dl ; 25/02/2021
8418      <1>      rcr     al, 1
8419      <1>      mov     [edi], al
8420      <1>      inc     dword [u.r0] ; +1
8421      <1>      m_pix_op_mix_8_1:
8422      <1>      inc     edi
8423      <1>      loop    m_pix_op_mix_8_0
8424      <1>      retn
8425      <1>      m_pix_op_mix_1:
8426      <1>      cmp     byte [v_bpp], 24 ; 24bpp
8427      <1>      ja      short m_pix_op_mix_3 ; 32bpp
8428      <1>      jb      short m_pix_op_mix_2 ; 16bpp
8429      <1>      ; 24 bit true colors
8430      <1>      jmp     short m_pix_op_mix_24
8431      <1>      m_pix_op_mix_24:
8432      <1>      ; 24 bit true colors
8433      <1>      mov     edx, eax ; new color
8434      <1>      and     edx, 0FFFFFFh
8435      <1>      m_pix_op_mix_24_0:
8436      <1>      lodsw
8437      <1>      shl     eax, 16
8438      <1>      lodsb
8439      <1>      rol     eax, 16
8440      <1>      cmp     eax, [maskcolor]
8441      <1>      je      short m_pix_op_mix_24_1 ; exclude
8442      <1>      add     eax, edx
8443      <1>      shr     eax, 1
8444      <1>      mov     [edi], ax
8445      <1>      shr     eax, 16
8446      <1>      mov     [edi+2], al
8447      <1>      add     dword [u.r0], 3 ; +3
8448      <1>      m_pix_op_mix_24_1:
8449      <1>      add     edi, 3 ; +3
8450      <1>      loop    m_pix_op_mix_24_0
8451      <1>      retn
8452      <1>      ; 65536 colors (16bpp)
8453      <1>      m_pix_op_mix_2:
8454      <1>      jmp     short m_pix_op_mix_16
8455      <1>      m_pix_op_mix_16:
8456      <1>      ; 16 bit colors (65536 colors)
8457      <1>      mov     edx, eax ; new color
8458      <1>      and     edx, 0FFFFh
8459      <1>      m_pix_op_mix_16_0:
8460      <1>      lodsw
8461      <1>      cmp     ax, [maskcolor]
8462      <1>      je      short m_pix_op_mix_16_1 ; exclude
8463      <1>      add     ax, dx
8464      <1>      rcr     ax, 1
8465      <1>      mov     [edi], ax
8466      <1>      add     dword [u.r0], 2 ; +2
8467      <1>      m_pix_op_mix_16_1:
8468      <1>      add     edi, 2 ; +2
8469      <1>      loop    m_pix_op_mix_16_0
8470      <1>      retn
8471      <1>      m_pix_op_mix_3:
8472      <1>      ; 32 bit true colors
8473      <1>      jmp     short m_pix_op_mix_32
8474      <1>      m_pix_op_mix_32:
8475      <1>      ; 32 bit true colors
8476      <1>      mov     edx, eax ; new color
8477      <1>      m_pix_op_mix_32_0:
8478      <1>      lodsd
8479      <1>      cmp     eax, [maskcolor]
8480      <1>      je      short m_pix_op_mix_32_1 ; exclude
8481      <1>      add     eax, edx
8482      <1>      ; 02/03/2021
8483      <1>      rcr     eax, 1
8484      <1>      mov     [edi], eax
8485      <1>      add     dword [u.r0], 4 ; +4
8486      <1>      m_pix_op_mix_32_1:
8487      <1>      add     edi, 4 ; +4
8488      <1>      loop    m_pix_op_mix_32_0
8489      <1>      retn
8490      <1>
8491      <1>      m_pix_op_mix_w:
8492      <1>      ; 06/02/2021
8493      <1>      ; MIX COLOR (MASKED, window)
8494      <1>      ;
8495      <1>      ; jump from pix_op_mix_w
8496      <1>      ;
8497      <1>      ; INPUT:
8498      <1>      ; ecx = bytes per row (to be applied)
8499      <1>      ; edx = screen width in bytes
8500      <1>      ; ebx = row count
8501      <1>      ; eax = color
8502      <1>      ;
8503      <1>      ; [maskcolor] = mask color (to be excluded)
8504      <1>      ;
8505      <1>      ; OUTPUT:
8506      <1>      ; [u.r0] will be > 0 if succesful
8507      <1>      ;
8508      <1>      ; window
8509      <1>      mov     edi, [v_str] ; LFB start address
8510      <1>      mov     esi, edi
8511      <1>
8512      <1>      cmp     byte [v_bpp], 8 ; 8bpp
8513      <1>      ja      short m_pix_op_mix_w_1
8514      <1>
8515      <1>      ; 256 colors (8bpp)
8516      <1>      mov     ebp, m_pix_op_mix_8
8517      <1>      jmp     short m_pix_op_mix_w_4
8518      <1>
8519      <1>      m_pix_op_mix_w_1:

```

```

8520 0000EC91 803D[799D0100]18 <1>    cmp     byte [v_bpp], 24 ; 24bpp
8521 0000EC98 7710 <1>    ja      short m_pix_op_mix_w_3 ; 32bpp
8522 0000EC9A 7207 <1>    jnb     short m_pix_op_mix_w_2 ; 16bpp
8523 <1>
8524 <1>    ; 24 bit true colors
8525 0000EC9C BD[13EC0000] <1>    mov     ebp, m_pix_op_mix_24
8526 0000ECA1 EB0C <1>    jmp     short m_pix_op_mix_w_4
8527 <1>
8528 <1>    ; 65536 colors (16bpp)
8529 <1> m_pix_op_mix_w_2:
8530 0000ECA3 BD[40EC0000] <1>    mov     ebp, m_pix_op_mix_16
8531 0000ECA8 EB05 <1>    jmp     short m_pix_op_mix_w_4
8532 <1>
8533 <1>    ; 32 bit true colors
8534 <1> m_pix_op_mix_w_3:
8535 0000ECAA BD[63EC0000] <1>    mov     ebp, m_pix_op_mix_32
8536 <1> m_pix_op_mix_w_4:
8537 0000ECAF E957FDFFFF <1>    jmp     m_pix_op_mix_w_x
8538 <1>
8539 <1> m_pix_op_and:
8540 <1>    ; 06/02/2021
8541 <1>    ; AND COLOR (MASKED, full screen)
8542 <1>
8543 <1>    ; jump from pix_op_and
8544 <1>
8545 <1>    ; INPUT:
8546 <1>    ;     eax = color (AL, AX, EAX)
8547 <1>    ;     ecx = [v_siz] ; display page pixel count
8548 <1>    ;     esi = edi = [v_mem] ; LFB start address
8549 <1>
8550 <1>    ;     [maskcolor] = mask color (to be excluded)
8551 <1>
8552 <1>    ; OUTPUT:
8553 <1>    ;     [u.r0] will be > 0 if succesful
8554 <1>
8555 <1>    ; Full screen
8556 <1> m_pix_op_and_0:
8557 0000ECB4 803D[799D0100]08 <1>    cmp     byte [v_bpp], 8 ; 8bpp
8558 0000ECBB 7717 <1>    ja      short m_pix_op_and_1
8559 <1>    ; 256 colors (8bpp)
8560 <1>    ; jmp     short m_pix_op_and_8
8561 <1> m_pix_op_and_8:
8562 <1>    ; 8 bit colors (256 colors)
8563 0000ECBD 88C2 <1>    mov     dl, al ; new color
8564 <1> m_pix_op_and_8_0:
8565 0000ECBF AC <1>    lodsb
8566 0000ECC0 3A05[8A9D0100] <1>    cmp     al, [maskcolor]
8567 0000ECC6 7408 <1>    je      short m_pix_op_and_8_1 ; exclude
8568 0000ECC8 2017 <1>    and     [edi], dl
8569 0000ECCA FF05[548E0100] <1>    inc     dword [u.r0] ; +1
8570 <1> m_pix_op_and_8_1:
8571 0000ECD0 47 <1>    inc     edi
8572 0000ECD1 E2EC <1>    loop    m_pix_op_and_8_0
8573 0000ECD3 C3 <1>    retn
8574 <1> m_pix_op_and_1:
8575 0000ECD4 803D[799D0100]18 <1>    cmp     byte [v_bpp], 24 ; 24bpp
8576 0000ECDB 774A <1>    ja      short m_pix_op_and_3 ; 32bpp
8577 0000ECDD 722B <1>    jnb     short m_pix_op_and_2 ; 16bpp
8578 <1>    ; 24 bit true colors
8579 <1>    ; jmp     short m_pix_op_and_24
8580 <1> m_pix_op_and_24:
8581 <1>    ; 24 bit true colors
8582 0000ECDF 89C2 <1>    mov     edx, eax ; new color
8583 <1>    ; and     edx, 0FFFFFFh
8584 <1> m_pix_op_and_24_0:
8585 0000ECE1 66AD <1>    lodsw
8586 0000ECE3 C1E010 <1>    shl     eax, 16
8587 0000ECE6 AC <1>    lodsb
8588 0000ECE7 C1C010 <1>    rol     eax, 16
8589 0000ECEA 3B05[8A9D0100] <1>    cmp     eax, [maskcolor]
8590 0000ECF0 7412 <1>    je      short m_pix_op_and_24_1 ; exclude
8591 0000ECF2 21D0 <1>    and     eax, edx
8592 0000ECF4 668907 <1>    mov     [edi], ax
8593 0000ECF7 C1E810 <1>    shr     eax, 16
8594 0000ECFA 884702 <1>    mov     [edi+2], al
8595 0000ECFD 8305[548E0100]03 <1>    add     dword [u.r0], 3 ; +3
8596 <1> m_pix_op_and_24_1:
8597 0000ED04 83C703 <1>    add     edi, 3 ; +3
8598 0000ED07 E2D8 <1>    loop    m_pix_op_and_24_0
8599 0000ED09 C3 <1>    retn
8600 <1>    ; 65536 colors (16bpp)
8601 <1> m_pix_op_and_2:
8602 <1>    ; jmp     short m_pix_op_and_16
8603 <1> m_pix_op_and_16:
8604 <1>    ; 16 bit colors (65536 colors)
8605 0000ED0A 89C2 <1>    mov     edx, eax ; new color
8606 <1>    ; and     edx, 0FFFFh
8607 <1> m_pix_op_and_16_0:
8608 0000ED0C 66AD <1>    lodsw
8609 0000ED0E 663B05[8A9D0100] <1>    cmp     ax, [maskcolor]
8610 0000ED15 740A <1>    je      short m_pix_op_and_16_1 ; exclude
8611 0000ED17 662117 <1>    and     [edi], dx
8612 0000ED1A 8305[548E0100]02 <1>    add     dword [u.r0], 2 ; +2
8613 <1> m_pix_op_and_16_1:
8614 0000ED21 83C702 <1>    add     edi, 2 ; +2
8615 0000ED24 E2E6 <1>    loop    m_pix_op_and_16_0
8616 0000ED26 C3 <1>    retn
8617 <1> m_pix_op_and_3:
8618 <1>    ; 32 bit true colors
8619 <1>    ; jmp     short m_pix_op_and_32
8620 <1> m_pix_op_and_32:
8621 <1>    ; 32 bit true colors
8622 0000ED27 89C2 <1>    mov     edx, eax ; new color
8623 <1> m_pix_op_and_32_0:
8624 0000ED29 AD <1>    lodsd
8625 0000ED2A 3B05[8A9D0100] <1>    cmp     eax, [maskcolor]
8626 0000ED30 7409 <1>    je      short m_pix_op_and_32_1 ; exclude
8627 0000ED32 2117 <1>    and     [edi], edx ; 25/02/2021
8628 0000ED34 8305[548E0100]04 <1>    add     dword [u.r0], 4 ; +4
8629 <1> m_pix_op_and_32_1:
8630 0000ED3B 83C704 <1>    add     edi, 4 ; +4
8631 0000ED3E E2E9 <1>    loop    m_pix_op_and_32_0
8632 0000ED40 C3 <1>    retn
8633 <1>
8634 <1> m_pix_op_and_w:
8635 <1>    ; 06/02/2021
8636 <1>    ; AND COLOR (MASKED, window)
8637 <1>
8638 <1>    ; jump from pix_op_and_w
8639 <1>
8640 <1>    ; INPUT:
8641 <1>    ;     ecx = bytes per row (to be applied)
8642 <1>    ;     edx = screen width in bytes
8643 <1>    ;     ebx = row count

```

```

8644      <1>      ;   eax = color
8645      <1>      ;
8646      <1>      ;   [maskcolor] = mask color (to be excluded)
8647      <1>      ;
8648      <1>      ;   OUTPUT:
8649      <1>      ;   [u.r0] will be > 0 if succesful
8650      <1>      ;
8651      <1>      ;   window
8652      <1>      ;mov   edi, [v_str] ; LFB start address
8653      <1>      ;mov   esi, edi
8654      <1>      ;
8655      0000ED41 803D[799D0100]08      <1>      cmp   byte [v_bpp], 8 ; 8bpp
8656      0000ED48 7707                  <1>      ja    short m_pix_op_and_w_1
8657      <1>      ;
8658      <1>      ;   256 colors (8bpp)
8659      0000ED4A BD[BDEC0000]          <1>      mov   ebp, m_pix_op_and_8
8660      0000ED4F EB1E                  <1>      jmp   short m_pix_op_and_w_4
8661      <1>      ;
8662      <1>      m_pix_op_and_w_1:
8663      0000ED51 803D[799D0100]18      <1>      cmp   byte [v_bpp], 24 ; 24bpp
8664      0000ED58 7710                  <1>      ja    short m_pix_op_and_w_3 ; 32bpp
8665      0000ED5A 7207                  <1>      jb    short m_pix_op_and_w_2 ; 16bpp
8666      <1>      ;
8667      <1>      ;   24 bit true colors
8668      0000ED5C BD[DFEC0000]          <1>      mov   ebp, m_pix_op_and_24
8669      0000ED61 EB0C                  <1>      jmp   short m_pix_op_and_w_4
8670      <1>      ;
8671      <1>      ;   65536 colors (16bpp)
8672      <1>      m_pix_op_and_w_2:
8673      0000ED63 BD[0AED0000]          <1>      mov   ebp, m_pix_op_and_16
8674      0000ED68 EB05                  <1>      jmp   short m_pix_op_and_w_4
8675      <1>      ;
8676      <1>      ;   32 bit true colors
8677      <1>      m_pix_op_and_w_3:
8678      0000ED6A BD[27ED0000]          <1>      mov   ebp, m_pix_op_and_32
8679      <1>      m_pix_op_and_w_4:
8680      0000ED6F E997FCFFFF          <1>      jmp   m_pix_op_and_w_x
8681      <1>      ;
8682      <1>      m_pix_op_or:
8683      <1>      ; 06/02/2021
8684      <1>      ; OR COLOR (MASKED, full screen)
8685      <1>      ;
8686      <1>      ; jump from pix_op_orc
8687      <1>      ;
8688      <1>      ;   INPUT:
8689      <1>      ;   eax = color (AL, AX, EAX)
8690      <1>      ;   ecx = [v_siz] ; display page pixel count
8691      <1>      ;   esi = edi = [v_mem] ; LFB start address
8692      <1>      ;
8693      <1>      ;   [maskcolor] = mask color (to be excluded)
8694      <1>      ;
8695      <1>      ;   OUTPUT:
8696      <1>      ;   [u.r0] will be > 0 if succesful
8697      <1>      ;
8698      <1>      ;   Full screen
8699      <1>      m_pix_op_or_0:
8700      0000ED74 803D[799D0100]08      <1>      cmp   byte [v_bpp], 8 ; 8bpp
8701      0000ED7B 7717                  <1>      ja    short m_pix_op_or_1
8702      <1>      ;
8703      <1>      ;   256 colors (8bpp)
8704      <1>      ; jmp   short m_pix_op_or_8
8705      <1>      m_pix_op_or_8:
8706      0000ED7D 88C2                  <1>      ; 8 bit colors (256 colors)
8707      <1>      mov   dl, al ; new color
8708      <1>      m_pix_op_or_8_0:
8709      0000ED7F AC                      <1>      lodsb
8710      0000ED80 3A05[8A9D0100]          <1>      cmp   al, [maskcolor]
8711      0000ED86 7408                  <1>      je    short m_pix_op_or_8_1 ; exclude
8712      0000ED88 0817                  <1>      or    [edi], dl
8713      0000ED8A FF05[548E0100]          <1>      inc   dword [u.r0] ; +1
8714      <1>      m_pix_op_or_8_1:
8715      0000ED90 47                      <1>      inc   edi
8716      0000ED91 E2EC                  <1>      loop  m_pix_op_or_8_0
8717      0000ED93 C3                      <1>      retn
8718      <1>      m_pix_op_or_1:
8719      0000ED94 803D[799D0100]18      <1>      cmp   byte [v_bpp], 24 ; 24bpp
8720      0000ED9B 774A                  <1>      ja    short m_pix_op_or_3 ; 32bpp
8721      0000ED9D 722B                  <1>      jb    short m_pix_op_or_2 ; 16bpp
8722      <1>      ;
8723      <1>      ;   24 bit true colors
8724      <1>      ; jmp   short m_pix_op_or_24
8725      <1>      m_pix_op_or_24:
8726      0000ED9F 89C2                  <1>      ; 24 bit true colors
8727      <1>      mov   edx, eax ; new color
8728      <1>      ; and   edx, 0FFFFFFh
8729      <1>      m_pix_op_or_24_0:
8730      0000EDA1 66AD                  <1>      lodsw
8731      0000EDA3 C1E010                <1>      shl   eax, 16
8732      0000EDA6 AC                      <1>      lodsb
8733      0000EDA7 C1C010                <1>      rol   eax, 16
8734      0000EDAA 3B05[8A9D0100]          <1>      cmp   eax, [maskcolor]
8735      0000EDB0 7412                  <1>      je    short m_pix_op_or_24_1 ; exclude
8736      0000EDB2 09D0                  <1>      or    eax, edx
8737      0000EDB4 668907                <1>      mov   [edi], ax
8738      0000EDB7 C1E810                <1>      shr   eax, 16
8739      0000EDBA 884702                <1>      mov   [edi+2], al
8740      0000EDBD 8305[548E0100]03      <1>      add   dword [u.r0], 3 ; +3
8741      <1>      m_pix_op_or_24_1:
8742      0000EDC4 83C703                <1>      add   edi, 3 ; +3
8743      0000EDC7 E2D8                  <1>      loop  m_pix_op_or_24_0
8744      0000EDC9 C3                      <1>      retn
8745      <1>      ; 65536 colors (16bpp)
8746      <1>      m_pix_op_or_2:
8747      <1>      ; jmp   short m_pix_op_or_16
8748      <1>      m_pix_op_or_16:
8749      0000EDCA 89C2                  <1>      ; 16 bit colors (65536 colors)
8750      <1>      mov   edx, eax ; new color
8751      <1>      ; and   edx, 0FFFFh
8752      <1>      m_pix_op_or_16_0:
8753      0000EDCC 66AD                  <1>      lodsw
8754      0000EDCE 663B05[8A9D0100]          <1>      cmp   ax, [maskcolor]
8755      0000EDD5 740A                  <1>      je    short m_pix_op_or_16_1 ; exclude
8756      0000EDD7 660917                <1>      or    [edi], dx
8757      0000EDDA 8305[548E0100]02      <1>      add   dword [u.r0], 2 ; +2
8758      <1>      m_pix_op_or_16_1:
8759      0000EDE1 83C702                <1>      add   edi, 2 ; +2
8760      0000EDE4 E2E6                  <1>      loop  m_pix_op_or_16_0
8761      0000EDE6 C3                      <1>      retn
8762      <1>      m_pix_op_or_3:
8763      <1>      ; 32 bit true colors
8764      <1>      ; jmp   short m_pix_op_or_32
8765      <1>      m_pix_op_or_32:
8766      <1>      ; 32 bit true colors
8767      0000EDE7 89C2                  <1>      mov   edx, eax ; new color
8768      <1>      m_pix_op_or_32_0:
8769      0000EDE9 AD                      <1>      lodsd

```

```

8768 0000EDEA 3B05[8A9D0100] <1> cmp     eax, [maskcolor]
8769 0000EDF0 7409 <1> je      short m_pix_op_or_32_1 ; exclude
8770 0000EDF2 0917 <1> or      [edi], edx ; 25/02/2021
8771 0000EDF4 8305[548E0100]04 <1> add     dword [u.r0], 4 ; +4
8772 <1> m_pix_op_or_32_1:
8773 0000EDFB 83C704 <1> add     edi, 4 ; +4
8774 0000EDFE E2E9 <1> loop    m_pix_op_or_32_0
8775 0000EE00 C3 <1> retn
8776 <1>
8777 <1> m_pix_op_or_w:
8778 <1> ; 06/02/2021
8779 <1> ; MIX COLOR (MASKED, window)
8780 <1> ;
8781 <1> ; jump from pix_op_or_w
8782 <1> ;
8783 <1> ; INPUT:
8784 <1> ; ecx = bytes per row (to be applied)
8785 <1> ; edx = screen width in bytes
8786 <1> ; ebx = row count
8787 <1> ; eax = color
8788 <1> ;
8789 <1> ; [maskcolor] = mask color (to be excluded)
8790 <1> ;
8791 <1> ; OUTPUT:
8792 <1> ; [u.r0] will be > 0 if succesful
8793 <1> ;
8794 <1> ; window
8795 <1> ; mov     edi, [v_str] ; LFB start address
8796 <1> ; mov     esi, edi
8797 <1> ;
8798 0000EE01 803D[799D0100]08 <1> cmp     byte [v_bpp], 8 ; 8bpp
8799 0000EE08 7707 <1> ja      short m_pix_op_or_w_1
8800 <1> ;
8801 <1> ; 256 colors (8bpp)
8802 0000EE0A BD[7DED0000] <1> mov     ebp, m_pix_op_or_8
8803 0000EE0F EB1E <1> jmp     short m_pix_op_or_w_4
8804 <1> ;
8805 <1> m_pix_op_or_w_1:
8806 0000EE11 803D[799D0100]18 <1> cmp     byte [v_bpp], 24 ; 24bpp
8807 0000EE18 7710 <1> ja      short m_pix_op_or_w_3 ; 32bpp
8808 0000EE1A 7207 <1> jb      short m_pix_op_or_w_2 ; 16bpp
8809 <1> ;
8810 <1> ; 24 bit true colors
8811 0000EE1C BD[9FED0000] <1> mov     ebp, m_pix_op_or_24
8812 0000EE21 EB0C <1> jmp     short m_pix_op_or_w_4
8813 <1> ;
8814 <1> ; 65536 colors (16bpp)
8815 <1> m_pix_op_or_w_2:
8816 0000EE23 BD[CAED0000] <1> mov     ebp, m_pix_op_or_16
8817 0000EE28 EB05 <1> jmp     short m_pix_op_or_w_4
8818 <1> ;
8819 <1> ; 32 bit true colors
8820 <1> m_pix_op_or_w_3:
8821 0000EE2A BD[E7ED0000] <1> mov     ebp, m_pix_op_or_32
8822 <1> m_pix_op_or_w_4:
8823 0000EE2F E9D7FBFFFF <1> jmp     m_pix_op_orc_w_x
8824 <1> ;
8825 <1> m_pix_op_xor:
8826 <1> ; 06/02/2021
8827 <1> ; XOR COLOR (MASKED, full screen)
8828 <1> ;
8829 <1> ; jump from pix_op_xor
8830 <1> ;
8831 <1> ; INPUT:
8832 <1> ; eax = color (AL, AX, EAX)
8833 <1> ; ecx = [v_siz] ; display page pixel count
8834 <1> ; esi = edi = [v_mem] ; LFB start address
8835 <1> ;
8836 <1> ; [maskcolor] = mask color (to be excluded)
8837 <1> ;
8838 <1> ; OUTPUT:
8839 <1> ; [u.r0] will be > 0 if succesful
8840 <1> ;
8841 <1> ; Full screen
8842 <1> m_pix_op_xor_0:
8843 0000EE34 803D[799D0100]08 <1> cmp     byte [v_bpp], 8 ; 8bpp
8844 0000EE3B 7717 <1> ja      short m_pix_op_xor_1
8845 <1> ; 256 colors (8bpp)
8846 <1> ; jmp     short m_pix_op_xor_8
8847 <1> m_pix_op_xor_8:
8848 <1> ; 8 bit colors (256 colors)
8849 0000EE3D 88C2 <1> mov     dl, al ; new color
8850 <1> m_pix_op_xor_8_0:
8851 0000EE3F AC <1> lodsb
8852 0000EE40 3A05[8A9D0100] <1> cmp     al, [maskcolor]
8853 0000EE46 7408 <1> je      short m_pix_op_xor_8_1 ; exclude
8854 0000EE48 3017 <1> xor     [edi], dl
8855 0000EE4A FF05[548E0100] <1> inc     dword [u.r0] ; +1
8856 <1> m_pix_op_xor_8_1:
8857 0000EE50 47 <1> inc     edi
8858 0000EE51 E2EC <1> loop    m_pix_op_xor_8_0
8859 0000EE53 C3 <1> retn
8860 <1> m_pix_op_xor_1:
8861 0000EE54 803D[799D0100]18 <1> cmp     byte [v_bpp], 24 ; 24bpp
8862 0000EE5B 774A <1> ja      short m_pix_op_xor_3 ; 32bpp
8863 0000EE5D 722B <1> jb      short m_pix_op_xor_2 ; 16bpp
8864 <1> ; 24 bit true colors
8865 <1> ; jmp     short m_pix_op_xor_24
8866 <1> m_pix_op_xor_24:
8867 <1> ; 24 bit true colors
8868 0000EE5F 89C2 <1> mov     edx, eax ; new color
8869 <1> ; and     edx, 0FFFFFFh
8870 <1> m_pix_op_xor_24_0:
8871 0000EE61 66AD <1> lodsw
8872 0000EE63 C1E010 <1> shl     eax, 16
8873 0000EE66 AC <1> lodsb
8874 0000EE67 C1C010 <1> rol     eax, 16
8875 0000EE6A 3B05[8A9D0100] <1> cmp     eax, [maskcolor]
8876 0000EE70 7412 <1> je      short m_pix_op_xor_24_1 ; exclude
8877 0000EE72 31D0 <1> xor     eax, edx
8878 0000EE74 668907 <1> mov     [edi], ax
8879 0000EE77 C1E810 <1> shr     eax, 16
8880 0000EE7A 884702 <1> mov     [edi+2], al
8881 0000EE7D 8305[548E0100]03 <1> add     dword [u.r0], 3 ; +3
8882 <1> m_pix_op_xor_24_1:
8883 0000EE84 83C703 <1> add     edi, 3 ; +3
8884 0000EE87 E2D8 <1> loop    m_pix_op_xor_24_0
8885 0000EE89 C3 <1> retn
8886 <1> ; 65536 colors (16bpp)
8887 <1> m_pix_op_xor_2:
8888 <1> ; jmp     short m_pix_op_xor_16
8889 <1> m_pix_op_xor_16:
8890 <1> ; 16 bit colors (65536 colors)
8891 0000EE8A 89C2 <1> mov     edx, eax ; new color

```

```

8892      <1>      ;and     edx, 0FFFFh
8893      <1> m_pix_op_xor_16_0:
8894      <1>      lodsw
8895      <1>      cmp     ax, [maskcolor]
8896      <1>      je      short m_pix_op_xor_16_1 ; exclude
8897      <1>      xor     [edi], dx
8898      <1>      add     dword [u.r0], 2 ; +2
8899      <1> m_pix_op_xor_16_1:
8900      <1>      add     edi, 2 ; +2
8901      <1>      loop   m_pix_op_xor_16_0
8902      <1>      retn
8903      <1> m_pix_op_xor_3:
8904      <1>      ; 32 bit true colors
8905      <1>      ; jmp     short m_pix_op_xor_32
8906      <1> m_pix_op_xor_32:
8907      <1>      ; 32 bit true colors
8908      <1>      mov     edx, eax ; new color
8909      <1> m_pix_op_xor_32_0:
8910      <1>      lodsd
8911      <1>      cmp     eax, [maskcolor]
8912      <1>      je      short m_pix_op_xor_32_1 ; exclude
8913      <1>      xor     [edi], edx ; 25/02/2021
8914      <1>      add     dword [u.r0], 4 ; +4
8915      <1> m_pix_op_xor_32_1:
8916      <1>      add     edi, 4 ; +4
8917      <1>      loop   m_pix_op_xor_32_0
8918      <1>      retn
8919      <1>
8920      <1> m_pix_op_xor_w:
8921      <1>      ; 06/02/2021
8922      <1>      ; XOR COLOR (MASKED, window)
8923      <1>      ;
8924      <1>      ; jump from pix_op_xor_w
8925      <1>      ;
8926      <1>      ; INPUT:
8927      <1>      ; ecx = bytes per row (to be applied)
8928      <1>      ; edx = screen width in bytes
8929      <1>      ; ebx = row count
8930      <1>      ; eax = color
8931      <1>      ;
8932      <1>      ; [maskcolor] = mask color (to be excluded)
8933      <1>      ;
8934      <1>      ; OUTPUT:
8935      <1>      ; [u.r0] will be > 0 if succesful
8936      <1>      ;
8937      <1>      ; window
8938      <1>      ; mov     edi, [v_str] ; LFB start address
8939      <1>      ; mov     esi, edi
8940      <1>
8941      <1>      cmp     byte [v_bpp], 8 ; 8bpp
8942      <1>      ja      short m_pix_op_xor_w_1
8943      <1>
8944      <1>      ; 256 colors (8bpp)
8945      <1>      mov     ebp, m_pix_op_xor_8
8946      <1>      jmp     short m_pix_op_xor_w_4
8947      <1>
8948      <1> m_pix_op_xor_w_1:
8949      <1>      cmp     byte [v_bpp], 24 ; 24bpp
8950      <1>      ja      short m_pix_op_xor_w_3 ; 32bpp
8951      <1>      jb      short m_pix_op_xor_w_2 ; 16bpp
8952      <1>
8953      <1>      ; 24 bit true colors
8954      <1>      mov     ebp, m_pix_op_xor_24
8955      <1>      jmp     short m_pix_op_xor_w_4
8956      <1>
8957      <1>      ; 65536 colors (16bpp)
8958      <1> m_pix_op_xor_w_2:
8959      <1>      mov     ebp, m_pix_op_xor_16
8960      <1>      jmp     short m_pix_op_xor_w_4
8961      <1>
8962      <1>      ; 32 bit true colors
8963      <1> m_pix_op_xor_w_3:
8964      <1>      mov     ebp, m_pix_op_xor_32
8965      <1> m_pix_op_xor_w_4:
8966      <1>      jmp     m_pix_op_xor_w_x
8967      <1>
8968      <1> m_pix_op_not:
8969      <1>      ; 06/02/2021
8970      <1>      ; NOT COLOR (MASKED, full screen)
8971      <1>      ;
8972      <1>      ; jump from pix_op_not
8973      <1>      ;
8974      <1>      ; INPUT:
8975      <1>      ; ecx = [v_siz] ; display page pixel count
8976      <1>      ; esi = edi = [v_mem] ; LFB start address
8977      <1>      ;
8978      <1>      ; [maskcolor] = mask color (to be excluded)
8979      <1>      ;
8980      <1>      ; OUTPUT:
8981      <1>      ; [u.r0] will be > 0 if succesful
8982      <1>      ;
8983      <1>      ; Full screen
8984      <1> m_pix_op_not_0:
8985      <1>      cmp     byte [v_bpp], 8 ; 8bpp
8986      <1>      ja      short m_pix_op_not_1
8987      <1>      ; 256 colors (8bpp)
8988      <1>      ; jmp     short m_pix_op_not_8
8989      <1> m_pix_op_not_8:
8990      <1>      ; 8 bit colors (256 colors)
8991      <1>      lodsb
8992      <1>      cmp     al, [maskcolor]
8993      <1>      je      short m_pix_op_not_8_1 ; exclude
8994      <1>      not     byte [edi]
8995      <1>      inc     dword [u.r0] ; +1
8996      <1> m_pix_op_not_8_1:
8997      <1>      inc     edi
8998      <1>      loop   m_pix_op_not_8
8999      <1>      retn
9000      <1> m_pix_op_not_1:
9001      <1>      cmp     byte [v_bpp], 24 ; 24bpp
9002      <1>      ja      short m_pix_op_not_3 ; 32bpp
9003      <1>      jb      short m_pix_op_not_2 ; 16bpp
9004      <1>      ; 24 bit true colors
9005      <1>      ; jmp     short m_pix_op_not_24
9006      <1> m_pix_op_not_24:
9007      <1>      ; 24 bit true colors
9008      <1>      lodsw
9009      <1>      shl     eax, 16
9010      <1>      lodsb
9011      <1>      rol     eax, 16
9012      <1>      cmp     eax, [maskcolor]
9013      <1>      je      short m_pix_op_not_24_1 ; exclude
9014      <1>      not     eax
9015      <1>      mov     [edi], ax

```



```

9016 0000EF33 C1E810      <1>      shr     eax, 16
9017 0000EF36 884702      <1>      mov     [edi+2], al
9018 0000EF39 8305[548E0100]03 <1>      add     dword [u.r0], 3 ; +3
9019                                <1> m_pix_op_not_24_1:
9020 0000EF40 83C703      <1>      add     edi, 3 ; +3
9021 0000EF43 E2D8      <1>      loop    m_pix_op_not_24
9022 0000EF45 C3      <1>      retn
9023                                <1> ; 65536 colors (16bpp)
9024                                <1> m_pix_op_not_2:
9025                                <1> ; jmp short m_pix_op_not_16
9026                                <1> m_pix_op_not_16:
9027                                <1> ; 16 bit colors (65536 colors)
9028 0000EF46 66AD      <1>      lodsw
9029 0000EF48 663B05[8A9D0100] <1>      cmp     ax, [maskcolor]
9030 0000EF4F 740A      <1>      je      short m_pix_op_not_16_1 ; exclude
9031 0000EF51 66F717      <1>      not     word [edi]
9032 0000EF54 8305[548E0100]02 <1>      add     dword [u.r0], 2 ; +2
9033                                <1> m_pix_op_not_16_1:
9034 0000EF5B 83C702      <1>      add     edi, 2 ; +2
9035 0000EF5E E2E6      <1>      loop    m_pix_op_not_16
9036 0000EF60 C3      <1>      retn
9037                                <1> m_pix_op_not_3:
9038                                <1> ; 32 bit true colors
9039                                <1> ; jmp short m_pix_op_not_32
9040                                <1> m_pix_op_not_32:
9041                                <1> ; 32 bit true colors
9042 0000EF61 AD      <1>      lodsd
9043 0000EF62 3B05[8A9D0100] <1>      cmp     eax, [maskcolor]
9044 0000EF68 7409      <1>      je      short m_pix_op_not_32_1 ; exclude
9045 0000EF6A F717      <1>      not     dword [edi]
9046 0000EF6C 8305[548E0100]04 <1>      add     dword [u.r0], 4 ; +4
9047                                <1> m_pix_op_not_32_1:
9048 0000EF73 83C704      <1>      add     edi, 4 ; +4
9049 0000EF76 E2E9      <1>      loop    m_pix_op_not_32
9050 0000EF78 C3      <1>      retn
9051                                <1>
9052                                <1> m_pix_op_not_w:
9053                                <1> ; 06/02/2021
9054                                <1> ; NOT COLOR (MASKED, window)
9055                                <1> ;
9056                                <1> ; jump from pix_op_not_w
9057                                <1> ;
9058                                <1> ; INPUT:
9059                                <1> ; ecx = bytes per row (to be applied)
9060                                <1> ; edx = screen width in bytes
9061                                <1> ; ebx = row count
9062                                <1> ;
9063                                <1> ; [maskcolor] = mask color (to be excluded)
9064                                <1> ;
9065                                <1> ; OUTPUT:
9066                                <1> ; [u.r0] will be > 0 if succesful
9067                                <1> ;
9068                                <1> ; window
9069                                <1> ; mov edi, [v_str] ; LFB start address
9070                                <1> ; mov esi, edi
9071                                <1>
9072 0000EF79 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
9073 0000EF80 7707      <1>      ja      short m_pix_op_not_w_1
9074                                <1>
9075                                <1> ; 256 colors (8bpp)
9076 0000EF82 BD[FDEE0000] <1>      mov     ebp, m_pix_op_not_8
9077 0000EF87 EB1E      <1>      jmp     short m_pix_op_not_w_4
9078                                <1>
9079                                <1> m_pix_op_not_w_1:
9080 0000EF89 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
9081 0000EF90 7710      <1>      ja      short m_pix_op_not_w_3 ; 32bpp
9082 0000EF92 7207      <1>      jb      short m_pix_op_not_w_2 ; 16bpp
9083                                <1>
9084                                <1> ; 24 bit true colors
9085 0000EF94 BD[1DEF0000] <1>      mov     ebp, m_pix_op_not_24
9086 0000EF99 EB0C      <1>      jmp     short m_pix_op_not_w_4
9087                                <1>
9088                                <1> ; 65536 colors (16bpp)
9089                                <1> m_pix_op_not_w_2:
9090 0000EF9B BD[46EF0000] <1>      mov     ebp, m_pix_op_not_16
9091 0000EFA0 EB05      <1>      jmp     short m_pix_op_not_w_4
9092                                <1>
9093                                <1> ; 32 bit true colors
9094                                <1> m_pix_op_not_w_3:
9095 0000EFA2 BD[61EF0000] <1>      mov     ebp, m_pix_op_not_32
9096                                <1> m_pix_op_not_w_4:
9097 0000EFA7 E95FFAFFFF <1>      jmp     m_pix_op_not_w_x
9098                                <1>
9099                                <1> m_pix_op_neg:
9100                                <1> ; 06/02/2021
9101                                <1> ; NEGATIVE COLOR (MASKED, full screen)
9102                                <1> ;
9103                                <1> ; jump from pix_op_neg
9104                                <1> ;
9105                                <1> ; INPUT:
9106                                <1> ; ecx = [v_siz] ; display page pixel count
9107                                <1> ; esi = edi = [v_mem] ; LFB start address
9108                                <1> ;
9109                                <1> ; [maskcolor] = mask color (to be excluded)
9110                                <1> ;
9111                                <1> ; OUTPUT:
9112                                <1> ; [u.r0] will be > 0 if succesful
9113                                <1> ;
9114                                <1> ; Full screen
9115                                <1> m_pix_op_neg_0:
9116 0000EFAC 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
9117 0000EFB3 7715      <1>      ja      short m_pix_op_neg_1
9118                                <1> ; 256 colors (8bpp)
9119                                <1> ; jmp short m_pix_op_neg_8
9120                                <1> m_pix_op_neg_8:
9121                                <1> ; 8 bit colors (256 colors)
9122 0000EFB5 AC      <1>      lodsb
9123 0000EFB6 3A05[8A9D0100] <1>      cmp     al, [maskcolor]
9124 0000EFBC 7408      <1>      je      short m_pix_op_neg_8_1 ; exclude
9125 0000EFBE F61F      <1>      neg     byte [edi]
9126 0000EFC0 FF05[548E0100] <1>      inc     dword [u.r0] ; +1
9127                                <1> m_pix_op_neg_8_1:
9128 0000EFC6 47      <1>      inc     edi
9129 0000EFC7 E2EC      <1>      loop    m_pix_op_neg_8
9130 0000EFC9 C3      <1>      retn
9131                                <1> m_pix_op_neg_1:
9132 0000EFCA 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
9133 0000EFD1 7746      <1>      ja      short m_pix_op_neg_3 ; 32bpp
9134 0000EFD3 7229      <1>      jb      short m_pix_op_neg_2 ; 16bpp
9135                                <1> ; 24 bit true colors
9136                                <1> ; jmp short m_pix_op_neg_24
9137                                <1> m_pix_op_neg_24:
9138                                <1> ; 24 bit true colors
9139 0000EFD5 66AD      <1>      lodsw

```

```

9140 0000EFD7 C1E010      <1>      shl      eax, 16
9141 0000EFDA AC          <1>      lodsb
9142 0000EFDB C1C010      <1>      rol      eax, 16
9143 0000EFDE 3B05[8A9D0100] <1>      cmp      eax, [maskcolor]
9144 0000EFE4 7412      <1>      je       short m_pix_op_neg_24_1 ; exclude
9145 0000EFE6 F7D8      <1>      neg      eax
9146 0000EFE8 668907      <1>      mov      [edi], ax
9147 0000EFEB C1E810      <1>      shr      eax, 16
9148 0000EFEE 884702      <1>      mov      [edi+2], al
9149 0000EFF1 8305[548E0100]03 <1>      add      dword [u.r0], 3 ; +3
9150                                <1> m_pix_op_neg_24_1:
9151 0000EFF8 83C703      <1>      add      edi, 3 ; +3
9152 0000EFFB E2D8      <1>      loop     m_pix_op_neg_24
9153 0000EFFD C3          <1>      retn
9154                                <1>      ; 65536 colors (16bpp)
9155                                <1> m_pix_op_neg_2:
9156                                <1>      ; jmp      short m_pix_op_neg_16
9157                                <1> m_pix_op_neg_16:
9158                                <1>      ; 16 bit colors (65536 colors)
9159 0000EFFE 66AD      <1>      lodsw
9160 0000F000 663B05[8A9D0100] <1>      cmp      ax, [maskcolor]
9161 0000F007 740A      <1>      je       short m_pix_op_neg_16_1 ; exclude
9162 0000F009 66F71F      <1>      neg      word [edi]
9163 0000F00C 8305[548E0100]02 <1>      add      dword [u.r0], 2 ; +2
9164                                <1> m_pix_op_neg_16_1:
9165 0000F013 83C702      <1>      add      edi, 2 ; +2
9166 0000F016 E2E6      <1>      loop     m_pix_op_neg_16
9167 0000F018 C3          <1>      retn
9168                                <1> m_pix_op_neg_3:
9169                                <1>      ; 32 bit true colors
9170                                <1>      ; jmp      short m_pix_op_neg_32
9171                                <1> m_pix_op_neg_32:
9172                                <1>      ; 32 bit true colors
9173 0000F019 AD          <1>      lodsd
9174 0000F01A 3B05[8A9D0100] <1>      cmp      eax, [maskcolor]
9175 0000F020 7409      <1>      je       short m_pix_op_neg_32_1 ; exclude
9176 0000F022 F71F      <1>      neg      dword [edi]
9177 0000F024 8305[548E0100]04 <1>      add      dword [u.r0], 4 ; +4
9178                                <1> m_pix_op_neg_32_1:
9179 0000F02B 83C704      <1>      add      edi, 4 ; +4
9180 0000F02E E2E9      <1>      loop     m_pix_op_neg_32
9181 0000F030 C3          <1>      retn
9182                                <1>
9183                                <1> m_pix_op_neg_w:
9184                                <1>      ; 06/02/2021
9185                                <1>      ; NEGATIVE COLOR (MASKED, window)
9186                                <1>
9187                                <1>      ; jump from pix_op_neg_w
9188                                <1>
9189                                <1>      ; INPUT:
9190                                <1>      ;     ecx = bytes per row (to be applied)
9191                                <1>      ;     edx = screen width in bytes
9192                                <1>      ;     ebx = row count
9193                                <1>
9194                                <1>      ; [maskcolor] = mask color (to be excluded)
9195                                <1>
9196                                <1>      ; OUTPUT:
9197                                <1>      ;     [u.r0] will be > 0 if succesful
9198                                <1>
9199                                <1>      ; window
9200                                <1>      ; mov      edi, [v_str] ; LFB start address
9201                                <1>      ; mov      esi, edi
9202                                <1>
9203 0000F031 803D[799D0100]08 <1>      cmp      byte [v_bpp], 8 ; 8bpp
9204 0000F038 7707      <1>      ja       short m_pix_op_neg_w_1
9205                                <1>
9206                                <1>      ; 256 colors (8bpp)
9207 0000F03A BD[B5EF0000] <1>      mov      ebp, m_pix_op_neg_8
9208 0000F03F EB1E      <1>      jmp      short m_pix_op_neg_w_4
9209                                <1>
9210                                <1> m_pix_op_neg_w_1:
9211 0000F041 803D[799D0100]18 <1>      cmp      byte [v_bpp], 24 ; 24bpp
9212 0000F048 7710      <1>      ja       short m_pix_op_neg_w_3 ; 32bpp
9213 0000F04A 7207      <1>      jb       short m_pix_op_neg_w_2 ; 16bpp
9214                                <1>
9215                                <1>      ; 24 bit true colors
9216 0000F04C BD[D5EF0000] <1>      mov      ebp, m_pix_op_neg_24
9217 0000F051 EB0C      <1>      jmp      short m_pix_op_neg_w_4
9218                                <1>
9219                                <1>      ; 65536 colors (16bpp)
9220                                <1> m_pix_op_neg_w_2:
9221 0000F053 BD[FEEF0000] <1>      mov      ebp, m_pix_op_neg_16
9222 0000F058 EB05      <1>      jmp      short m_pix_op_neg_w_4
9223                                <1>
9224                                <1>      ; 32 bit true colors
9225                                <1> m_pix_op_neg_w_3:
9226 0000F05A BD[19F00000] <1>      mov      ebp, m_pix_op_neg_32
9227                                <1> m_pix_op_neg_w_4:
9228 0000F05F E9A7F9FFFF <1>      jmp      m_pix_op_neg_w_x
9229                                <1>
9230                                <1> m_pix_op_inc:
9231                                <1>      ; 06/02/2021
9232                                <1>      ; INCREASE COLOR (MASKED, full screen)
9233                                <1>
9234                                <1>      ; jump from pix_op_inc
9235                                <1>
9236                                <1>      ; INPUT:
9237                                <1>      ;     ecx = [v_siz] ; display page pixel count
9238                                <1>      ;     esi = edi = [v_mem] ; LFB start address
9239                                <1>
9240                                <1>      ; [maskcolor] = mask color (to be excluded)
9241                                <1>
9242                                <1>      ; OUTPUT:
9243                                <1>      ;     [u.r0] will be > 0 if succesful
9244                                <1>
9245                                <1>      ; Full screen
9246                                <1> m_pix_op_inc_0:
9247 0000F064 803D[799D0100]08 <1>      cmp      byte [v_bpp], 8 ; 8bpp
9248 0000F06B 7719      <1>      ja       short m_pix_op_inc_1
9249                                <1>      ; 256 colors (8bpp)
9250                                <1>      ; jmp      short m_pix_op_inc_8
9251                                <1> m_pix_op_inc_8:
9252                                <1>      ; 8 bit colors (256 colors)
9253 0000F06D AC          <1>      lodsb
9254 0000F06E 3A05[8A9D0100] <1>      cmp      al, [maskcolor]
9255 0000F074 740C      <1>      je       short m_pix_op_inc_8_1 ; exclude
9256 0000F076 FE07      <1>      inc      byte [edi]
9257 0000F078 7502      <1>      jnz      short m_pix_op_inc_8_0
9258 0000F07A FE0F      <1>      dec      byte [edi]
9259                                <1> m_pix_op_inc_8_0:
9260 0000F07C FF05[548E0100] <1>      inc      dword [u.r0] ; +1
9261                                <1> m_pix_op_inc_8_1:
9262 0000F082 47          <1>      inc      edi
9263 0000F083 E2E8      <1>      loop     m_pix_op_inc_8

```

```

9264 0000F085 C3          <1>      retn
9265                      <1> m_pix_op_inc_1:
9266 0000F086 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
9267 0000F08D 7752          <1>      ja      short m_pix_op_inc_3 ; 32bpp
9268 0000F08F 7230          <1>      jb      short m_pix_op_inc_2 ; 16bpp
9269                      <1>      ; 24 bit true colors
9270                      <1>      ; jmp     short m_pix_op_inc_24
9271                      <1> m_pix_op_inc_24:
9272                      <1>      ; 24 bit true colors
9273 0000F091 66AD          <1>      lodsw
9274 0000F093 C1E010        <1>      shl     eax, 16
9275 0000F096 AC              <1>      lodsb
9276 0000F097 C1C010        <1>      rol     eax, 16
9277 0000F09A 3805[8A9D0100] <1>      cmp     eax, [maskcolor]
9278 0000F0A0 7419          <1>      je      short m_pix_op_inc_24_1 ; exclude
9279 0000F0A2 40              <1>      inc     eax
9280 0000F0A3 3DFFFFFFF00    <1>      cmp     eax, 0FFFFFFFh
9281 0000F0A8 7601          <1>      jna     short m_pix_op_inc_24_0
9282 0000F0AA 48              <1>      dec     eax
9283                      <1> m_pix_op_inc_24_0:
9284 0000F0AB 668907        <1>      mov     [edi], ax
9285 0000F0AE C1E810        <1>      shr     eax, 16
9286 0000F0B1 884702        <1>      mov     [edi+2], al
9287 0000F0B4 8305[548E0100]03 <1>      add     dword [u.r0], 3 ; +3
9288                      <1> m_pix_op_inc_24_1:
9289 0000F0BB 83C703        <1>      add     edi, 3 ; +3
9290 0000F0BE E2D1          <1>      loop    m_pix_op_inc_24
9291 0000F0C0 C3              <1>      retn
9292                      <1>      ; 65536 colors (16bpp)
9293                      <1> m_pix_op_inc_2:
9294                      <1>      ; jmp     short m_pix_op_inc_16
9295                      <1> m_pix_op_inc_16:
9296                      <1>      ; 16 bit colors (65536 colors)
9297 0000F0C1 66AD          <1>      lodsw
9298 0000F0C3 663B05[8A9D0100] <1>      cmp     ax, [maskcolor]
9299 0000F0CA 740F          <1>      je      short m_pix_op_inc_16_1 ; exclude
9300 0000F0CC 66FF07        <1>      inc     word [edi]
9301 0000F0CF 7503          <1>      jnz     short m_pix_op_inc_16_0
9302 0000F0D1 66FF0F        <1>      dec     word [edi]
9303                      <1> m_pix_op_inc_16_0:
9304 0000F0D4 8305[548E0100]02 <1>      add     dword [u.r0], 2 ; +2
9305                      <1> m_pix_op_inc_16_1:
9306 0000F0DB 83C702        <1>      add     edi, 2 ; +2
9307 0000F0DE E2E1          <1>      loop    m_pix_op_inc_16
9308 0000F0E0 C3              <1>      retn
9309                      <1> m_pix_op_inc_3:
9310                      <1>      ; 32 bit true colors
9311                      <1>      ; jmp     short m_pix_op_inc_32
9312                      <1> m_pix_op_inc_32:
9313                      <1>      ; 32 bit true colors
9314 0000F0E1 AD              <1>      lodsd
9315 0000F0E2 3B05[8A9D0100] <1>      cmp     eax, [maskcolor]
9316 0000F0E8 740D          <1>      je      short m_pix_op_inc_32_1 ; exclude
9317 0000F0EA FF07          <1>      inc     dword [edi]
9318 0000F0EC 7502          <1>      jnz     short m_pix_op_inc_32_0
9319 0000F0EE FF0F          <1>      dec     dword [edi]
9320                      <1> m_pix_op_inc_32_0:
9321 0000F0F0 8305[548E0100]04 <1>      add     dword [u.r0], 4 ; +4
9322                      <1> m_pix_op_inc_32_1:
9323 0000F0F7 83C704        <1>      add     edi, 4 ; +4
9324 0000F0FA E2E5          <1>      loop    m_pix_op_inc_32
9325 0000F0FC C3              <1>      retn
9326                      <1>
9327                      <1> m_pix_op_inc_w:
9328                      <1>      ; 06/02/2021
9329                      <1>      ; INCREASE COLOR (MASKED, window)
9330                      <1>      ;
9331                      <1>      ; jump from pix_op_inc_w
9332                      <1>      ;
9333                      <1>      ; INPUT:
9334                      <1>      ;     ecx = bytes per row (to be applied)
9335                      <1>      ;     edx = screen width in bytes
9336                      <1>      ;     ebx = row count
9337                      <1>      ;
9338                      <1>      ; [maskcolor] = mask color (to be excluded)
9339                      <1>      ;
9340                      <1>      ; OUTPUT:
9341                      <1>      ;     [u.r0] will be > 0 if succesful
9342                      <1>      ;
9343                      <1>      ; window
9344                      <1>      ; mov     edi, [v_str] ; LFB start address
9345                      <1>      ; mov     esi, edi
9346                      <1>
9347 0000F0FD 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
9348 0000F104 7707          <1>      ja      short m_pix_op_inc_w_1
9349                      <1>
9350                      <1>      ; 256 colors (8bpp)
9351 0000F106 BD[6DF00000]    <1>      mov     ebp, m_pix_op_inc_8
9352 0000F10B EB1E          <1>      jmp     short m_pix_op_inc_w_4
9353                      <1>
9354                      <1> m_pix_op_inc_w_1:
9355 0000F10D 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
9356 0000F114 7710          <1>      ja      short m_pix_op_inc_w_3 ; 32bpp
9357 0000F116 7207          <1>      jb      short m_pix_op_inc_w_2 ; 16bpp
9358                      <1>
9359                      <1>      ; 24 bit true colors
9360 0000F118 BD[91F00000]    <1>      mov     ebp, m_pix_op_inc_24
9361 0000F11D EB0C          <1>      jmp     short m_pix_op_inc_w_4
9362                      <1>
9363                      <1>      ; 65536 colors (16bpp)
9364                      <1> m_pix_op_inc_w_2:
9365 0000F11F BD[C1F00000]    <1>      mov     ebp, m_pix_op_inc_16
9366 0000F124 EB05          <1>      jmp     short m_pix_op_inc_w_4
9367                      <1>
9368                      <1>      ; 32 bit true colors
9369                      <1> m_pix_op_inc_w_3:
9370 0000F126 BD[E1F00000]    <1>      mov     ebp, m_pix_op_inc_32
9371                      <1> m_pix_op_inc_w_4:
9372 0000F12B E9DBF8FFFF      <1>      jmp     m_pix_op_inc_w_x
9373                      <1>
9374                      <1> m_pix_op_dec:
9375                      <1>      ; 06/02/2021
9376                      <1>      ; DECREASE COLOR (MASKED, full screen)
9377                      <1>      ;
9378                      <1>      ; jump from pix_op_dec
9379                      <1>      ;
9380                      <1>      ; INPUT:
9381                      <1>      ;     ecx = [v_siz] ; display page pixel count
9382                      <1>      ;     esi = edi = [v_mem] ; LFB start address
9383                      <1>      ;
9384                      <1>      ; [maskcolor] = mask color (to be excluded)
9385                      <1>      ;
9386                      <1>      ; OUTPUT:
9387                      <1>      ;     [u.r0] will be > 0 if succesful

```

```

9388 <1>
9389 <1> ; Full screen
9390 <1> m_pix_op_dec_0:
9391 0000F130 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
9392 0000F137 7719 <1> ja short m_pix_op_dec_1
9393 <1> ; 256 colors (8bpp)
9394 <1> ; jmp short m_pix_op_dec_8
9395 <1> m_pix_op_dec_8:
9396 <1> ; 8 bit colors (256 colors)
9397 0000F139 AC <1> lodsb
9398 0000F13A 3A05[8A9D0100] <1> cmp al, [maskcolor]
9399 0000F140 740C <1> je short m_pix_op_dec_8_1 ; exclude
9400 0000F142 FE0F <1> dec byte [edi]
9401 0000F144 7902 <1> jns short m_pix_op_dec_8_0
9402 0000F146 FE07 <1> inc byte [edi]
9403 <1> m_pix_op_dec_8_0:
9404 0000F148 FF05[548E0100] <1> inc dword [u.r0] ; +1
9405 <1> m_pix_op_dec_8_1:
9406 0000F14E 47 <1> inc edi
9407 0000F14F E2E8 <1> loop m_pix_op_dec_8
9408 0000F151 C3 <1> retn
9409 <1> m_pix_op_dec_1:
9410 0000F152 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
9411 0000F159 774D <1> ja short m_pix_op_dec_3 ; 32bpp
9412 0000F15B 722B <1> jb short m_pix_op_dec_2 ; 16bpp
9413 <1> ; 24 bit true colors
9414 <1> ; jmp short m_pix_op_dec_24
9415 <1> m_pix_op_dec_24:
9416 <1> ; 24 bit true colors
9417 0000F15D 66AD <1> lodsw
9418 0000F15F C1E010 <1> shl eax, 16
9419 0000F162 AC <1> lodsb
9420 0000F163 C1C010 <1> rol eax, 16
9421 0000F166 3B05[8A9D0100] <1> cmp eax, [maskcolor]
9422 0000F16C 7414 <1> je short m_pix_op_dec_24_1 ; exclude
9423 0000F16E 48 <1> dec eax
9424 0000F16F 7901 <1> jns short m_pix_op_dec_24_0
9425 0000F171 40 <1> inc eax
9426 <1> m_pix_op_dec_24_0:
9427 0000F172 668907 <1> mov [edi], ax
9428 0000F175 C1E810 <1> shr eax, 16
9429 0000F178 884702 <1> mov [edi+2], al
9430 0000F17B 8305[548E0100]03 <1> add dword [u.r0], 3 ; +3
9431 <1> m_pix_op_dec_24_1:
9432 0000F182 83C703 <1> add edi, 3 ; +3
9433 0000F185 E2D6 <1> loop m_pix_op_dec_24
9434 0000F187 C3 <1> retn
9435 <1> ; 65536 colors (16bpp)
9436 <1> m_pix_op_dec_2:
9437 <1> ; jmp short m_pix_op_dec_16
9438 <1> m_pix_op_dec_16:
9439 <1> ; 16 bit colors (65536 colors)
9440 0000F188 66AD <1> lodsw
9441 0000F18A 663B05[8A9D0100] <1> cmp ax, [maskcolor]
9442 0000F191 740F <1> je short m_pix_op_dec_16_1 ; exclude
9443 0000F193 66FF0F <1> dec word [edi]
9444 0000F196 7903 <1> jns short m_pix_op_dec_16_0
9445 0000F198 66FF07 <1> inc word [edi]
9446 <1> m_pix_op_dec_16_0:
9447 0000F19B 8305[548E0100]02 <1> add dword [u.r0], 2 ; +2
9448 <1> m_pix_op_dec_16_1:
9449 0000F1A2 83C702 <1> add edi, 2 ; +2
9450 0000F1A5 E2E1 <1> loop m_pix_op_dec_16
9451 0000F1A7 C3 <1> retn
9452 <1> m_pix_op_dec_3:
9453 <1> ; 32 bit true colors
9454 <1> ; jmp short m_pix_op_dec_32
9455 <1> m_pix_op_dec_32:
9456 <1> ; 32 bit true colors
9457 0000F1A8 AD <1> lodsd
9458 0000F1A9 3B05[8A9D0100] <1> cmp eax, [maskcolor]
9459 0000F1AF 740D <1> je short m_pix_op_dec_32_1 ; exclude
9460 0000F1B1 FF0F <1> dec dword [edi]
9461 0000F1B3 7902 <1> jns short m_pix_op_dec_32_0
9462 0000F1B5 FF07 <1> inc dword [edi]
9463 <1> m_pix_op_dec_32_0:
9464 0000F1B7 8305[548E0100]04 <1> add dword [u.r0], 4 ; +4
9465 <1> m_pix_op_dec_32_1:
9466 0000F1BE 83C704 <1> add edi, 4 ; +4
9467 0000F1C1 E2E5 <1> loop m_pix_op_dec_32
9468 0000F1C3 C3 <1> retn
9469 <1>
9470 <1> m_pix_op_dec_w:
9471 <1> ; 06/02/2021
9472 <1> ; DECREASE COLOR (MASKED, window)
9473 <1> ;
9474 <1> ; jump from pix_op_dec_w
9475 <1> ;
9476 <1> ; INPUT:
9477 <1> ; ecx = bytes per row (to be applied)
9478 <1> ; edx = screen width in bytes
9479 <1> ; ebx = row count
9480 <1> ;
9481 <1> ; [maskcolor] = mask color (to be excluded)
9482 <1> ;
9483 <1> ; OUTPUT:
9484 <1> ; [u.r0] will be > 0 if succesful
9485 <1> ;
9486 <1> ; window
9487 <1> ; mov edi, [v_str] ; LFB start address
9488 <1> ; mov esi, edi
9489 <1> ;
9490 0000F1C4 803D[799D0100]08 <1> cmp byte [v_bpp], 8 ; 8bpp
9491 0000F1CB 7707 <1> ja short m_pix_op_dec_w_1
9492 <1> ;
9493 <1> ; 256 colors (8bpp)
9494 0000F1CD BD[39F10000] <1> mov ebp, m_pix_op_dec_8
9495 0000F1D2 EB1E <1> jmp short m_pix_op_dec_w_4
9496 <1> ;
9497 <1> m_pix_op_dec_w_1:
9498 0000F1D4 803D[799D0100]18 <1> cmp byte [v_bpp], 24 ; 24bpp
9499 0000F1DB 7710 <1> ja short m_pix_op_dec_w_3 ; 32bpp
9500 0000F1DD 7207 <1> jb short m_pix_op_dec_w_2 ; 16bpp
9501 <1> ;
9502 <1> ; 24 bit true colors
9503 0000F1DF BD[5DF10000] <1> mov ebp, m_pix_op_dec_24
9504 0000F1E4 EB0C <1> jmp short m_pix_op_dec_w_4
9505 <1> ;
9506 <1> ; 65536 colors (16bpp)
9507 <1> m_pix_op_dec_w_2:
9508 0000F1E6 BD[88F10000] <1> mov ebp, m_pix_op_dec_16
9509 0000F1EB EB05 <1> jmp short m_pix_op_dec_w_4
9510 <1> ;
9511 <1> ; 32 bit true colors

```

```

9512      <1> m_pix_op_dec_w_3:
9513 0000F1ED BD[A8F10000] <1> mov     ebp, m_pix_op_dec_32
9514      <1> m_pix_op_dec_w_4:
9515 0000F1F2 E914F8FFFF <1> jmp     m_pix_op_dec_w_x
9516      <1>
9517      <1> sysvideo_39:
9518      <1> ; 15/02/2021
9519      <1> ; 07/02/2021, 08/02/2021
9520      <1> ; 03/01/2021, 04/01/2021
9521      <1> ; 23/11/2020
9522      <1> ; BH = 3
9523      <1> ; PIXEL READ/WRITE
9524      <1>
9525      <1> ; 07/02/2021
9526      <1> ; 04/01/2021 (TRDOS 386 v2.0.3)
9527 0000F1F7 80FB03 <1> cmp     bl, 3
9528 0000F1FA 761A <1> jna     short sysvideo_39_1
9529      <1> ; 07/02/2021
9530 0000F1FC 80FB06 <1> cmp     bl, 6
9531 0000F1FF 7705 <1> ja      short sysvideo_39_0
9532 0000F201 E91A010000 <1> jmp     sysvideo_39_31
9533      <1> sysvideo_39_0:
9534      <1> ; error
9535 0000F206 B3FF <1> mov     bl, 0FFh
9536 0000F208 8B2D[508E0100] <1> mov     ebp, [u.usp] ; ebp points to user's registers
9537 0000F20E 895D10 <1> mov     [ebp+16], ebx ; EBX
9538 0000F211 E968D8FFFF <1> jmp     sysret
9539      <1> sysvideo_39_1:
9540 0000F216 803D[1E680000]FF <1> cmp     byte [CRT_MODE], 0FFh
9541 0000F21D 7312 <1> jnb     short sysvideo_39_2 ; SVGA (VESA VBE) video mode
9542      <1>
9543      <1> ; Std VGA or CGA mode
9544 0000F21F 81C200000A00 <1> add     edx, 0A0000h
9545 0000F225 72DF <1> jc      short sysvideo_39_0
9546 0000F227 81FAFFFF0A00 <1> cmp     edx, 0AFFFFFh
9547 0000F22D 77D7 <1> ja      short sysvideo_39_0
9548 0000F22F EB1E <1> jmp     short sysvideo_39_3 ; 8bpp
9549      <1>
9550      <1> sysvideo_39_2:
9551      <1> ; use current vbe (svga) video mode
9552      <1>
9553      <1> ; get LFB address
9554 0000F231 A1[1C9D0100] <1> mov     eax, [LFB_ADDR] ; [LFB_Info+LFBINFO.LFB_addr]
9555 0000F236 09C0 <1> or      eax, eax
9556 0000F238 74CC <1> jz      short sysvideo_39_0
9557 0000F23A 3B15[209D0100] <1> cmp     edx, [LFB_SIZE]
9558 0000F240 73C4 <1> jnb     short sysvideo_39_0
9559      <1>
9560 0000F242 01C2 <1> add     edx, eax
9561      <1> ;jc     short sysvideo_39_0
9562      <1>
9563      <1> ; Pixel read/write in VESA VBE (2/3) video mode
9564      <1> ; Video memory at Linear Frame Buffer base address
9565      <1>
9566 0000F244 8A3D[289D0100] <1> mov     bh, [LFB_Info+LFBINFO.bpp]
9567      <1>
9568 0000F24A 80FF08 <1> cmp     bh, 8 ; 8bpp
9569 0000F24D 775D <1> ja      short sysvideo_39_17
9570      <1>
9571      <1> ; 8 bits per pixel
9572      <1> sysvideo_39_3:
9573 0000F24F 80FB01 <1> cmp     bl, 1 ; 1 = write pixel
9574 0000F252 7406 <1> je      short sysvideo_39_5
9575 0000F254 7712 <1> ja      short sysvideo_39_8
9576      <1> sysvideo_39_4:
9577      <1> ; read pixel (8bpp)
9578 0000F256 8A02 <1> mov     al, [edx]
9579      <1> ;mov    [u.r0], al
9580      <1> ;jmp    sysret
9581 0000F258 EB04 <1> jmp     short sysvideo_39_7
9582      <1> sysvideo_39_5:
9583      <1> ; write pixel (8bpp)
9584 0000F25A 88C8 <1> mov     al, cl
9585      <1> sysvideo_39_6:
9586 0000F25C 8802 <1> mov     [edx], al
9587      <1> sysvideo_39_7:
9588 0000F25E A2[548E0100] <1> mov     [u.r0], al
9589 0000F263 E916D8FFFF <1> jmp     sysret
9590      <1> sysvideo_39_8:
9591 0000F268 80FB03 <1> cmp     bl, 3 ; mix
9592 0000F26B 7208 <1> jb      short sysvideo_39_9
9593      <1> ; mix pixel colors (8bpp)
9594 0000F26D 8A02 <1> mov     al, [edx]
9595 0000F26F 00C8 <1> add     al, cl
9596 0000F271 D0D8 <1> rcr     al, 1
9597 0000F273 EBE7 <1> jmp     short sysvideo_39_6
9598      <1> sysvideo_39_9:
9599      <1> ; swap pixel colors (8bpp)
9600 0000F275 88C8 <1> mov     al, cl
9601 0000F277 8602 <1> xchg    [edx], al
9602 0000F279 EBE3 <1> jmp     short sysvideo_39_7
9603      <1>
9604      <1> ; 16 bits per pixel
9605      <1> sysvideo_39_10:
9606 0000F27B 80FB01 <1> cmp     bl, 1 ; 1 = write pixel
9607 0000F27E 7406 <1> je      short sysvideo_39_12
9608 0000F280 7714 <1> ja      short sysvideo_39_15
9609      <1> sysvideo_39_11:
9610      <1> ; read pixel (16bpp)
9611 0000F282 8B02 <1> mov     eax, [edx]
9612      <1> ;mov    [u.r0], ax
9613      <1> ;jmp    sysret
9614 0000F284 EB05 <1> jmp     short sysvideo_39_14
9615      <1> sysvideo_39_12:
9616      <1> ; write pixel (16bpp)
9617 0000F286 89C8 <1> mov     eax, ecx
9618      <1> sysvideo_39_13:
9619 0000F288 668902 <1> mov     [edx], ax
9620      <1> sysvideo_39_14:
9621 0000F28B 66A3[548E0100] <1> mov     [u.r0], ax
9622 0000F291 E9E8D7FFFF <1> jmp     sysret
9623      <1> sysvideo_39_15:
9624 0000F296 80FB03 <1> cmp     bl, 3 ; mix
9625 0000F299 720A <1> jb      short sysvideo_39_16
9626      <1> ; mix pixel colors (16bpp)
9627 0000F29B 8B02 <1> mov     eax, [edx]
9628 0000F29D 6601C8 <1> add     ax, cx
9629 0000F2A0 66D1D8 <1> rcr     ax, 1
9630 0000F2A3 EBE3 <1> jmp     short sysvideo_39_13
9631      <1> sysvideo_39_16:
9632      <1> ; swap pixel colors (16bpp)
9633 0000F2A5 89C8 <1> mov     eax, ecx
9634 0000F2A7 668702 <1> xchg    [edx], ax
9635 0000F2AA EBD F <1> jmp     short sysvideo_39_14

```

```

9636 <1> sysvideo_39_17:
9637 0000F2AC 80FF18 <1> cmp bh, 24
9638 0000F2AF 7743 <1> ja short sysvideo_39_24
9639 0000F2B1 72C8 <1> jb short sysvideo_39_10
9640 <1>
9641 <1> ; 24 bits per pixel
9642 0000F2B3 81E1FFFFFFF00 <1> and ecx, 0FFFFFFFh
9643 0000F2B9 80FB01 <1> cmp bl, 1 ; 1 = write pixel
9644 0000F2BC 7406 <1> je short sysvideo_39_19
9645 0000F2BE 7712 <1> ja short sysvideo_39_22
9646 <1> sysvideo_39_18:
9647 <1> ; read pixel (24bpp)
9648 0000F2C0 8B02 <1> mov eax, [edx]
9649 <1> ;and eax, 0FFFFFFFh
9650 <1> ;mov [u.r0], eax
9651 <1> ;jmp sysret
9652 0000F2C2 EB04 <1> jmp short sysvideo_39_21
9653 <1> sysvideo_39_19:
9654 <1> ; write pixel (24bpp)
9655 0000F2C4 89C8 <1> mov eax, ecx
9656 <1> sysvideo_39_20:
9657 <1> ;and eax, 0FFFFFFFh
9658 0000F2C6 8902 <1> mov [edx], eax
9659 <1> sysvideo_39_21:
9660 0000F2C8 A3[548E0100] <1> mov [u.r0], eax
9661 0000F2CD E9ACD7FFFF <1> jmp sysret
9662 <1> sysvideo_39_22:
9663 0000F2D2 80FB03 <1> cmp bl, 3 ; mix
9664 0000F2D5 720D <1> jb short sysvideo_39_23
9665 <1> ; mix pixel colors (24bpp)
9666 0000F2D7 8B02 <1> mov eax, [edx]
9667 0000F2D9 25FFFFFFF00 <1> and eax, 0FFFFFFFh
9668 <1> ;and ecx, 0FFFFFFFh
9669 0000F2DE 01C8 <1> add eax, ecx
9670 0000F2E0 D1D8 <1> rcr eax, 1
9671 0000F2E2 EBE2 <1> jmp short sysvideo_39_20
9672 <1> sysvideo_39_23:
9673 <1> ; swap pixel colors (24bpp)
9674 0000F2E4 89C8 <1> mov eax, ecx
9675 <1> ;and eax, 0FFFFFFFh
9676 0000F2E6 668702 <1> xchg [edx], ax
9677 0000F2E9 C1C810 <1> ror eax, 16
9678 0000F2EC 884202 <1> mov [edx+2], al
9679 0000F2EF C1C010 <1> rol eax, 16
9680 0000F2F2 EBD4 <1> jmp short sysvideo_39_21
9681 <1>
9682 <1> ; 32 bits per pixel
9683 <1> sysvideo_39_24:
9684 0000F2F4 80FB01 <1> cmp bl, 1 ; 1 = write pixel
9685 0000F2F7 7406 <1> je short sysvideo_39_26
9686 0000F2F9 7712 <1> ja short sysvideo_39_29
9687 <1> sysvideo_39_25:
9688 <1> ; read pixel (32bpp)
9689 0000F2FB 8B02 <1> mov eax, [edx]
9690 <1> ;mov [u.r0], eax
9691 <1> ;jmp sysret
9692 0000F2FD EB04 <1> jmp short sysvideo_39_28
9693 <1> sysvideo_39_26:
9694 <1> ; write pixel (32bpp)
9695 0000F2FF 89C8 <1> mov eax, ecx
9696 <1> sysvideo_39_27:
9697 0000F301 8902 <1> mov [edx], eax
9698 <1> sysvideo_39_28:
9699 0000F303 A3[548E0100] <1> mov [u.r0], eax
9700 0000F308 E971D7FFFF <1> jmp sysret
9701 <1> sysvideo_39_29:
9702 0000F30D 80FB03 <1> cmp bl, 3 ; mix
9703 0000F310 7208 <1> jb short sysvideo_39_30
9704 <1> ; mix pixel colors (32bpp)
9705 0000F312 8B02 <1> mov eax, [edx]
9706 0000F314 01C8 <1> add eax, ecx
9707 0000F316 D1D8 <1> rcr eax, 1
9708 0000F318 EBE7 <1> jmp short sysvideo_39_27
9709 <1> sysvideo_39_30:
9710 <1> ; swap pixel colors (32bpp)
9711 0000F31A 89C8 <1> mov eax, ecx
9712 0000F31C 8702 <1> xchg [edx], eax
9713 0000F31E EBE3 <1> jmp short sysvideo_39_28
9714 <1>
9715 <1> sysvideo_39_31:
9716 <1> ; 06/03/2021
9717 <1> ; 08/02/2021
9718 <1> ; 07/02/2021
9719 <1> ; BL = 4 -> read pixels from user defined positions
9720 <1> ; BL = 5 -> write single color pixels to user defined pos.
9721 <1> ; BL = 6 -> write multi color pixels to user defined pos.
9722 <1> ; ECX = color (CL, CX, ECX)
9723 <1> ; EDX = number of pixels
9724 <1> ; ESI = user buffer contains dword pixel positions
9725 <1> ; (and dword colors for BL input = 6)
9726 <1> ; EDI = user's pixel color buff (destination) for BL = 4
9727 <1>
9728 0000F320 890D[8A9D0100] <1> mov [maskcolor], ecx
9729 0000F326 89D5 <1> mov ebp, edx ; number of pixels
9730 0000F328 803D[1E680000]FF <1> cmp byte [CRT_MODE], 0FFh ; SVGA flag
9731 0000F32F 7317 <1> jnb short sysvideo_39_33 ; SVGA (VESA VBE mode)
9732 <1> ; Standard VGA mode
9733 0000F331 8900000100 <1> mov ecx, 65536 ; video page size (maximum)
9734 0000F336 39CA <1> cmp edx, ecx
9735 0000F338 7709 <1> ja short sysvideo_39_32 ; abnormal value !
9736 0000F33A B800000A00 <1> mov eax, 0A0000h ; video page start address
9737 0000F33F B708 <1> mov bh, 8 ; 8 bits per pixel (256 colors)
9738 0000F341 EB35 <1> jmp short sysvideo_39_34
9739 <1> sysvideo_39_32:
9740 <1> ; nonsense! (edx has abnormal value)
9741 0000F343 E936D7FFFF <1> jmp sysret
9742 <1> sysvideo_39_33:
9743 <1> ; 06/03/2021
9744 0000F348 8A3D[289D0100] <1> mov bh, [LFB_Info+LFBINFO.bpp]
9745 0000F34E 80FF08 <1> cmp bh, 8
9746 0000F351 7412 <1> je short sysvideo_39_81 ; 8bpp
9747 0000F353 89D0 <1> mov eax, edx
9748 0000F355 80FF10 <1> cmp bh, 16
9749 0000F358 7409 <1> je short sysvideo_39_80 ; 16bpp
9750 0000F35A D1E2 <1> shl edx, 1
9751 0000F35C 80FF20 <1> cmp bh, 32
9752 0000F35F 7502 <1> jne short sysvideo_39_80 ; 24bpp
9753 0000F361 D1E0 <1> shl eax, 1
9754 <1> sysvideo_39_80:
9755 0000F363 01C2 <1> add edx, eax
9756 <1> ; edx = number of bytes
9757 <1> sysvideo_39_81:
9758 <1> ; get LFB address
9759 0000F365 A1[1C9D0100] <1> mov eax, [LFB_ADDR] ; [LFB_Info+LFBINFO.LFB_addr]

```

```

9760 0000F36A 09C0      <1>      or      eax, eax
9761 0000F36C 74D5      <1>      jz      short sysvideo_39_32 ; LFB is not ready !
9762 0000F36E 8B0D[209D0100]    <1>      mov     ecx, [LFB_SIZE]
9763 0000F374 39CA      <1>      cmp     edx, ecx
9764 0000F376 77CB      <1>      ja      short sysvideo_39_32 ; abnormal value !
9765                                     <1>
9766                                     <1>      ; 02/03/2021
9767                                     <1>      ; 08/02/2021
9768                                     <1>      ;mov     ebp, edx ; pixel count
9769                                     <1>      ;shl     ebp, 2 ; byte count (pixel pos: 4 bytes)
9770                                     <1>
9771                                     <1>      ; 06/03/2021
9772                                     <1>      ; bits per pixel (pixel color size)
9773                                     <1>      ;mov     bh, [LFB_Info+LFBINFO.bpp]
9774 sysvideo_39_34:
9775 0000F378 C1E502    <1>      shl     ebp, 2 ; 15/02/2021 (byte count)
9776 0000F37B A3[7A9D0100]    <1>      mov     [v_mem], eax ; Save video page start address
9777 0000F380 88DE      <1>      mov     dh, bl ; sub function
9778                                     <1>      ; 06/03/2021
9779 0000F382 883D[799D0100]    <1>      mov     [v_bpp], bh ; bits per pixel (color size)
9780                                     <1>      ;mov     ebx, [LFB_SIZE]
9781 0000F388 89CB      <1>      mov     ebx, ecx ; [LFB_SIZE]
9782                                     <1>
9783 0000F38A B900080000    <1>      mov     ecx, 2048
9784 0000F38F 39CD      <1>      cmp     ebp, ecx
9785 0000F391 7302      <1>      jnb     short sysvideo_39_35
9786 0000F393 89E9      <1>      mov     ecx, ebp ; fix to requested byte count
9787 sysvideo_39_35:
9788 0000F395 80FE04    <1>      cmp     dh, 4 ; 08/02/2021
9789                                     <1>      ;cmp     bl, 4 ; read pixels from user defined positions
9790 0000F398 7605      <1>      jna     short sysvideo_39_36
9791 0000F39A E9B2000000    <1>      jmp     sysvideo_39_52
9792                                     <1>      ; 08/02/2021
9793                                     <1>      ;mov     [buffer8], edi ; user's destination buff addr
9794 sysvideo_39_36:
9795                                     <1>      ; 08/02/2021
9796                                     <1>      ; read pixel positions
9797                                     <1>      ; as defined in user's source buffer
9798 0000F39F 893D[929D0100]    <1>      mov     [buffer8], edi ; user's destination buff addr
9799 0000F3A5 BF00760900    <1>      mov     edi, VBE3SAVERESTOREBLOCK ; kernel buffer for
9800                                     <1>      ; 2028 byte data
9801                                     <1>      ; esi = user's source buffer for pixel positions
9802                                     <1>      ; ecx = byte count
9803 0000F3AA E8A6170000    <1>      call    transfer_from_user_buffer
9804 0000F3AF 7292      <1>      jc      short sysvideo_39_32 ; error
9805                                     <1>      ; ecx = transfer count (bytes)
9806                                     <1>
9807 0000F3B1 57      <1>      push     edi ; *
9808 0000F3B2 56      <1>      push     esi ; **
9809 0000F3B3 51      <1>      push     ecx ; ***
9810                                     <1>
9811 0000F3B4 89FE      <1>      mov     esi, edi ; kernel buffer
9812 0000F3B6 8B15[7A9D0100]    <1>      mov     edx, [v_mem] ; video memory
9813 0000F3BC C1E902    <1>      shr     ecx, 2 ; pixel count (within buffer capacity)
9814                                     <1>
9815 0000F3BF 803D[799D0100]08    <1>      cmp     byte [v_bpp], 8 ; 8bpp
9816 0000F3C6 7753      <1>      ja      short sysvideo_39_49
9817 sysvideo_39_37:
9818                                     <1>      ; 8bpp
9819 0000F3C8 AD      <1>      lodsd
9820 0000F3C9 39D8      <1>      cmp     eax, ebx ; < [LFB_SIZE]
9821 0000F3CB 7309      <1>      jnb     short sysvideo_39_39
9822 0000F3CD 0FB60402    <1>      movzx   eax, byte [edx+eax]
9823 sysvideo_39_38:
9824 0000F3D1 AB      <1>      stosd
9825 0000F3D2 E2F4      <1>      loop   sysvideo_39_37
9826 0000F3D4 EB49      <1>      jmp     short sysvideo_39_50
9827 sysvideo_39_39:
9828                                     <1>      ; write black color for improper positions
9829 0000F3D6 31C0      <1>      xor     eax, eax
9830 0000F3D8 EBF7      <1>      jmp     short sysvideo_39_38
9831 sysvideo_39_40:
9832 0000F3DA 803D[799D0100]18    <1>      cmp     byte [v_bpp], 24 ; 24bpp
9833 0000F3E1 772A      <1>      ja      short sysvideo_39_47 ; 32bpp
9834 0000F3E3 7216      <1>      jb      short sysvideo_39_44 ; 16bpp
9835 sysvideo_39_41:
9836                                     <1>      ; 24bpp
9837 0000F3E5 AD      <1>      lodsd
9838 0000F3E6 39D8      <1>      cmp     eax, ebx ; < [LFB_SIZE]
9839 0000F3E8 730D      <1>      jnb     short sysvideo_39_43
9840 0000F3EA 8B0402    <1>      mov     eax, [edx+eax]
9841 0000F3ED 25FFFFFF00    <1>      and     eax, 0FFFFFFh
9842 sysvideo_39_42:
9843 0000F3F2 AB      <1>      stosd
9844 0000F3F3 E2F0      <1>      loop   sysvideo_39_41
9845 0000F3F5 EB28      <1>      jmp     short sysvideo_39_50
9846 sysvideo_39_43:
9847                                     <1>      ; write black color for improper positions
9848 0000F3F7 31C0      <1>      xor     eax, eax
9849 0000F3F9 EBF7      <1>      jmp     short sysvideo_39_42
9850 sysvideo_39_44:
9851                                     <1>      ; 16bpp
9852 0000F3FB AD      <1>      lodsd
9853 0000F3FC 39D8      <1>      cmp     eax, ebx ; < [LFB_SIZE]
9854 0000F3FE 7309      <1>      jnb     short sysvideo_39_46
9855 0000F400 0FB70402    <1>      movzx   eax, word [edx+eax]
9856 sysvideo_39_45:
9857 0000F404 AB      <1>      stosd
9858 0000F405 E2F4      <1>      loop   sysvideo_39_44
9859 0000F407 EB16      <1>      jmp     short sysvideo_39_50
9860 sysvideo_39_46:
9861                                     <1>      ; write black color for improper positions
9862 0000F409 31C0      <1>      xor     eax, eax
9863 0000F40B EBF7      <1>      jmp     short sysvideo_39_45
9864 sysvideo_39_47:
9865                                     <1>      ; 32bpp
9866 0000F40D AD      <1>      lodsd
9867 0000F40E 39D8      <1>      cmp     eax, ebx ; < [LFB_SIZE]
9868 0000F410 7309      <1>      jnb     short sysvideo_39_49
9869 0000F412 0FB70402    <1>      movzx   eax, word [edx+eax]
9870 sysvideo_39_48:
9871 0000F416 AB      <1>      stosd
9872 0000F417 E2F4      <1>      loop   sysvideo_39_47
9873 0000F419 EB04      <1>      jmp     short sysvideo_39_50
9874 sysvideo_39_49:
9875                                     <1>      ; write black color for improper positions
9876 0000F41B 31C0      <1>      xor     eax, eax
9877 0000F41D EBF7      <1>      jmp     short sysvideo_39_48
9878 sysvideo_39_50:
9879 0000F41F 59      <1>      pop     ecx ; transfer count in bytes
9880 0000F420 5E      <1>      pop     esi ; ** ; kernel buffer
9881                                     <1>      ;mov     esi, VBE3SAVERESTOREBLOCK ; kernel buffer for
9882                                     <1>      ; 2048 byte data
9883 0000F421 8B3D[929D0100]    <1>      mov     edi, [buffer8]

```

```

9884      <1>      ; edi = user's destination buffer for pixel colors
9885      <1>      ; ecx = byte count
9886      0000F427 E8DF160000 <1>      call    transfer_to_user_buffer
9887      0000F42C 5E <1>      pop     esi ; *
9888      0000F42D 7266 <1>      jc     short sysvideo_39_56 ; error
9889      <1>      ; ecx = transfer count (bytes)
9890      0000F42F 89C8 <1>      mov     eax, ecx
9891      0000F431 C1E802 <1>      shr     eax, 2
9892      0000F434 0105[548E0100] <1>      add     [u.r0], eax ; transfer count (in pixels)
9893      <1>
9894      0000F43A 29CD <1>      sub     ebp, ecx
9895      0000F43C 7657 <1>      jna     short sysvideo_39_56 ; completed/finished
9896      0000F43E 01CE <1>      add     esi, ecx ; next position in source buffer
9897      <1>      ;add    [buffer8], ecx ; next pos in destination buff
9898      0000F440 01CF <1>      add     edi, ecx
9899      0000F442 66B90008 <1>      mov     cx, 2048 ; new count, limit: kernel buff size
9900      0000F446 39CD <1>      cmp     ebp, ecx ; remain >= limit ?
9901      0000F448 7302 <1>      jnb     short sysvideo_39_51 ; yes
9902      0000F44A 89E9 <1>      mov     ecx, ebp ; fix byte count to remain bytes
9903      <1>      sysvideo_39_51:
9904      0000F44C E94EFFFFFF <1>      jmp     sysvideo_39_36
9905      <1>
9906      <1>      sysvideo_39_52:
9907      0000F451 80FE05 <1>      cmp     dh, 5 ; 08/02/2021
9908      <1>      ;cmp    bl, 5 ; write pixels to user defined positions
9909      0000F454 7605 <1>      jna     short sysvideo_39_53
9910      0000F456 E9A1000000 <1>      jmp     sysvideo_39_66
9911      <1>      sysvideo_39_53:
9912      <1>      ; single color pixel writing
9913      0000F45B BF00760900 <1>      mov     edi, VBE3SAVERESTOREBLOCK ; kernel buffer for
9914      <1>      ; 2028 byte data
9915      <1>      ; esi = user's source buffer for pixel positions
9916      <1>      ; ecx = byte count
9917      0000F460 E8F0160000 <1>      call    transfer_from_user_buffer
9918      0000F465 722E <1>      jc     short sysvideo_39_56 ; error
9919      <1>      ; ecx = transfer count (bytes)
9920      <1>
9921      <1>      ; write pixels by using (user) defined positions
9922      <1>      ; ecx = byte count (1,2,3,4 times pixel count)
9923      <1>      ; edi = system buffer address
9924      <1>
9925      0000F467 56 <1>      push    esi ; *
9926      0000F468 51 <1>      push    ecx ; **
9927      <1>
9928      0000F469 89FE <1>      mov     esi, edi
9929      0000F46B 8B3D[7A9D0100] <1>      mov     edi, [v_mem]
9930      <1>
9931      <1>      ; 08/02/2021
9932      0000F471 C1E902 <1>      shr     ecx, 2 ; pixel count
9933      0000F474 8B15[8A9D0100] <1>      mov     edx, [maskcolor]
9934      <1>      ;mov    ebx, [v_siz]
9935      <1>
9936      0000F47A 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
9937      0000F481 7717 <1>      ja     short sysvideo_39_57
9938      <1>      sysvideo_39_54:
9939      <1>      ; 8bpp
9940      0000F483 AD <1>      lodsd
9941      0000F484 39D8 <1>      cmp     eax, ebx ; < [v_siz]
9942      0000F486 7309 <1>      jnb     short sysvideo_39_55
9943      0000F488 8B1407 <1>      mov     [edi+eax], dl
9944      <1>      ; 06/03/2021
9945      0000F48B FF05[548E0100] <1>      inc     dword [u.r0]
9946      <1>      sysvideo_39_55:
9947      0000F491 E2F0 <1>      loop    sysvideo_39_54
9948      0000F493 EB50 <1>      jmp     short sysvideo_39_64
9949      <1>      sysvideo_39_56:
9950      0000F495 E9E4D5FFFF <1>      jmp     sysret
9951      <1>      sysvideo_39_57:
9952      0000F49A 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
9953      0000F4A1 7732 <1>      ja     short sysvideo_39_62 ; 32bpp
9954      0000F4A3 721D <1>      jb     short sysvideo_39_60 ; 16bpp
9955      <1>      sysvideo_39_58:
9956      <1>      ; 24bpp
9957      0000F4A5 AD <1>      lodsd
9958      0000F4A6 39D8 <1>      cmp     eax, ebx ; < [v_siz]
9959      0000F4A8 7314 <1>      jnb     short sysvideo_39_59
9960      0000F4AA 8B1407 <1>      mov     [edi+eax], dl
9961      0000F4AD 40 <1>      inc     eax
9962      0000F4AE C1CA08 <1>      ror     edx, 8
9963      0000F4B1 66891407 <1>      mov     [edi+eax], dx
9964      0000F4B5 C1C208 <1>      rol     edx, 8
9965      0000F4B8 FF05[548E0100] <1>      inc     dword [u.r0]
9966      <1>      sysvideo_39_59:
9967      0000F4BE E2E5 <1>      loop    sysvideo_39_58
9968      0000F4C0 EB23 <1>      jmp     short sysvideo_39_64
9969      <1>      sysvideo_39_60:
9970      <1>      ; 16bpp
9971      0000F4C2 AD <1>      lodsd
9972      0000F4C3 39D8 <1>      cmp     eax, ebx ; < [v_siz]
9973      0000F4C5 730A <1>      jnb     short sysvideo_39_61
9974      0000F4C7 66891407 <1>      mov     [edi+eax], dx
9975      0000F4CB FF05[548E0100] <1>      inc     dword [u.r0]
9976      <1>      sysvideo_39_61:
9977      0000F4D1 E2EF <1>      loop    sysvideo_39_60
9978      0000F4D3 EB10 <1>      jmp     short sysvideo_39_64
9979      <1>      sysvideo_39_62:
9980      <1>      ; 32bpp
9981      0000F4D5 AD <1>      lodsd
9982      0000F4D6 39D8 <1>      cmp     eax, ebx ; < [v_siz]
9983      0000F4D8 7309 <1>      jnb     short sysvideo_39_63
9984      0000F4DA 8B1407 <1>      mov     [edi+eax], edx
9985      0000F4DD FF05[548E0100] <1>      inc     dword [u.r0]
9986      <1>      sysvideo_39_63:
9987      0000F4E3 E2F0 <1>      loop    sysvideo_39_62
9988      <1>      sysvideo_39_64:
9989      0000F4E5 59 <1>      pop     ecx ; **
9990      0000F4E6 5E <1>      pop     esi ; *
9991      0000F4E7 29CD <1>      sub     ebp, ecx
9992      0000F4E9 76AA <1>      jna     short sysvideo_39_56
9993      0000F4EB 01CE <1>      add     esi, ecx
9994      0000F4ED 66B90008 <1>      mov     cx, 2048
9995      0000F4F1 39CD <1>      cmp     ebp, ecx
9996      0000F4F3 7302 <1>      jnb     short sysvideo_39_65
9997      0000F4F5 89E9 <1>      mov     ecx, ebp
9998      <1>      sysvideo_39_65:
9999      0000F4F7 E95FFFFFFF <1>      jmp     sysvideo_39_53
10000      <1>
10001      <1>      sysvideo_39_66:
10002      <1>      ; 15/02/2021
10003      0000F4FC D1E5 <1>      shl     ebp, 1 ; 8 bytes per pixel (position&color)
10004      <1>      sysvideo_39_67:
10005      0000F4FE 66B90008 <1>      mov     cx, 2048
10006      0000F502 39CD <1>      cmp     ebp, ecx
10007      0000F504 7302 <1>      jnb     short sysvideo_39_68

```



```

10008 0000F506 89E9      <1>      mov     ecx, ebp
10009                      <1> sysvideo_39_68:
10010                      <1> ; multi colors pixel writing
10011 0000F508 BF00760900 <1>      mov     edi, VBE3SAVERESTOREBLOCK ; kernel buffer for
10012                      <1> ; 2048 byte data
10013                      <1> ; esi = user's source buffer for pixel positions
10014                      <1> ; ecx = byte count
10015 0000F50D E843160000 <1>      call    transfer_from_user_buffer
10016 0000F512 7281      <1>      jc      short sysvideo_39_56 ; error
10017                      <1> ; ecx = transfer count
10018                      <1>
10019                      <1> ; write pixels & colors as defined in user buffer
10020                      <1> ; ecx = byte count (2,4,6,8 times pixel count)
10021                      <1> ; edi = system buffer address
10022                      <1>
10023 0000F514 56          <1>      push    esi ; **
10024 0000F515 51          <1>      push    ecx ; *
10025                      <1>
10026 0000F516 89FE      <1>      mov     esi, edi
10027 0000F518 8B3D[7A9D0100] <1>      mov     edi, [v_mem]
10028                      <1>
10029                      <1> ; 08/02/2021
10030 0000F51E C1E903      <1>      shr     ecx, 3 ; pixel count
10031                      <1>
10032                      <1> ;mov     ebx, [v_siz]
10033                      <1>
10034 0000F521 803D[799D0100]08 <1>      cmp     byte [v_bpp], 8 ; 8bpp
10035 0000F528 7715      <1>      ja      short sysvideo_39_71
10036                      <1> sysvideo_39_69:
10037                      <1> ; 8bpp
10038 0000F52A AD          <1>      lodsd    ; position
10039 0000F52B 89C2      <1>      mov     edx, eax
10040 0000F52D AD          <1>      lodsd    ; color
10041 0000F52E 39DA      <1>      cmp     edx, ebx ; < [v_siz]
10042 0000F530 7309      <1>      jnb     short sysvideo_39_70
10043 0000F532 8B0417      <1>      mov     [edi+edx], al
10044                      <1> ; 06/03/2021
10045 0000F535 FF05[548E0100] <1>      inc     dword [u.r0]
10046                      <1> sysvideo_39_70:
10047 0000F53B E2ED      <1>      loop    sysvideo_39_69
10048 0000F53D EB51      <1>      jmp     short sysvideo_39_78
10049                      <1> sysvideo_39_71:
10050 0000F53F 803D[799D0100]18 <1>      cmp     byte [v_bpp], 24 ; 24bpp
10051 0000F546 7735      <1>      ja      short sysvideo_39_76 ; 32bpp
10052 0000F548 721D      <1>      jb      short sysvideo_39_74 ; 16bpp
10053                      <1> sysvideo_39_72:
10054                      <1> ; 24bpp
10055 0000F54A AD          <1>      lodsd    ; position
10056 0000F54B 89C2      <1>      mov     edx, eax
10057 0000F54D AD          <1>      lodsd    ; color
10058 0000F54E 39DA      <1>      cmp     edx, ebx ; < [v_siz]
10059 0000F550 7311      <1>      jnb     short sysvideo_39_73
10060 0000F552 8B0417      <1>      mov     [edi+edx], al
10061 0000F555 42          <1>      inc     edx
10062 0000F556 C1E808      <1>      shr     eax, 8
10063 0000F559 66890417 <1>      mov     [edi+edx], ax
10064 0000F55D FF05[548E0100] <1>      inc     dword [u.r0]
10065                      <1> sysvideo_39_73:
10066 0000F563 E2E5      <1>      loop    sysvideo_39_72
10067 0000F565 EB29      <1>      jmp     short sysvideo_39_78
10068                      <1> sysvideo_39_74:
10069                      <1> ; 16bpp
10070 0000F567 AD          <1>      lodsd    ; position
10071 0000F568 89C2      <1>      mov     edx, eax
10072 0000F56A AD          <1>      lodsd    ; color
10073 0000F56B 39DA      <1>      cmp     edx, ebx ; < [v_siz]
10074 0000F56D 730A      <1>      jnb     short sysvideo_39_75
10075 0000F56F 66890417 <1>      mov     [edi+edx], ax
10076 0000F573 FF05[548E0100] <1>      inc     dword [u.r0]
10077                      <1> sysvideo_39_75:
10078 0000F579 E2EC      <1>      loop    sysvideo_39_74
10079 0000F57B EB13      <1>      jmp     short sysvideo_39_78
10080                      <1> sysvideo_39_76:
10081                      <1> ; 32bpp
10082 0000F57D AD          <1>      lodsd    ; position
10083 0000F57E 89C2      <1>      mov     edx, eax
10084 0000F580 AD          <1>      lodsd    ; color
10085 0000F581 39DA      <1>      cmp     edx, ebx ; < [v_siz]
10086 0000F583 7309      <1>      jnb     short sysvideo_39_77
10087 0000F585 8B0417      <1>      mov     [edi+edx], eax
10088 0000F588 FF05[548E0100] <1>      inc     dword [u.r0]
10089                      <1> sysvideo_39_77:
10090 0000F58E E2ED      <1>      loop    sysvideo_39_76
10091                      <1> sysvideo_39_78:
10092 0000F590 59          <1>      pop     ecx ; *
10093 0000F591 5E          <1>      pop     esi ; **
10094                      <1>
10095 0000F592 29CD      <1>      sub     ebp, ecx
10096 0000F594 762A      <1>      jna     short sysvideo_39_79
10097 0000F596 01CE      <1>      add     esi, ecx
10098 0000F598 E961FFFFFF <1>      jmp     sysvideo_39_67
10099                      <1> ;sysvideo_39_79:
10100                      <1> ; jmp     sysret
10101                      <1>
10102                      <1> sysvideo_16:
10103                      <1> ; 11/08/2022
10104                      <1> ; 23/07/2022
10105                      <1> ; 06/03/2021
10106                      <1> ; 23/11/2020
10107 0000F59D 80FF04      <1>      cmp     bh, 4
10108                      <1> ;jb     sysvideo_39 ; bh = 3, pixel r/w
10109                      <1> ;ja     short sysvideo_17
10110                      <1> ; 23/07/2022
10111 0000F5A0 7407      <1>      je      short sysvideo_16_0
10112 0000F5A2 7721      <1>      ja      short sysvideo_17
10113 0000F5A4 E94EFCFFFF <1>      jmp     sysvideo_39
10114                      <1> sysvideo_16_0:
10115                      <1> ; BH = 4
10116                      <1> ; Direct User Access for CGA video memory.
10117                      <1> ; Setup user's page tables for direct access to 0B8000h.
10118                      <1> ;
10119                      <1> ; Permission checks are not implemented yet !
10120                      <1> ; (11/07/2016)
10121                      <1>
10122 0000F5A9 B800800B00 <1>      mov     eax, 0B8000h
10123                      <1> ;mov     ecx, 8 ; 8 pages (8*4K=32K)
10124                      <1> ; 11/08/2022
10125 0000F5AE 31C9      <1>      xor     ecx, ecx
10126 0000F5B0 B108      <1>      mov     cl, 8
10127 0000F5B2 89C3      <1>      mov     ebx, eax ; 12/05/2017 ; virtual = physical
10128 0000F5B4 E80A6AFFFF <1>      call    direct_memory_access
10129                      <1> ;jc     sysret
10130 0000F5B9 7205      <1>      jc      short sysvideo_39_79 ; 06/03/2021
10131                      <1> ; eax = 0B8000h if there is not an error

```

```

10132 0000F5BB A3[548E0100] <1> mov [u.r0], eax
10133 <1> sysvideo_39_79: ; 08/01/2021
10134 0000F5C0 E9B9D4FFFF <1> jmp sysret
10135 <1>
10136 <1> sysvideo_17:
10137 <1> ; 23/07/2022
10138 <1> ; 23/12/2020
10139 <1> ; 11/12/2020
10140 <1> ; 10/12/2020
10141 <1> ; 23/11/2020
10142 0000F5C5 80FF06 <1> cmp bh, 6
10143 0000F5C8 7424 <1> je short sysvideo_17_0 ; 23/07/2022
10144 0000F5CA 7205 <1> jb short sysvideo_18 ; 23/07/2022
10145 0000F5CC E95E010000 <1> jmp sysvideo_20 ; ja
10146 <1>
10147 <1> ; 23/07/2022
10148 <1> sysvideo_18:
10149 <1> ; BH = 5
10150 <1> ; Direct User Access for VGA video memory.
10151 <1> ; Setup user's page tables for direct access to 0A0000h.
10152 <1> ;
10153 <1> ; Permission checks are not implemented yet !
10154 <1> ; (11/07/2016)
10155 <1>
10156 0000F5D1 B800000A00 <1> mov eax, 0A0000h
10157 0000F5D6 B910000000 <1> mov ecx, 16 ; 16 pages (16*4K=64K)
10158 0000F5DB 89C3 <1> mov ebx, eax ; 12/05/2017 ; virtual = physical
10159 0000F5DD E8E169FFFF <1> call direct_memory_access
10160 <1> ;jc sysret
10161 <1> ; 23/07/2022
10162 0000F5E2 7205 <1> jc short sysvideo_18_0
10163 <1> ; eax = 0A0000h if there is not an error
10164 0000F5E4 A3[548E0100] <1> mov [u.r0], eax
10165 <1> sysvideo_18_0:
10166 0000F5E9 E990D4FFFF <1> jmp sysret
10167 <1>
10168 <1> sysvideo_17_0:
10169 <1> ; BH = 6
10170 <1> ; Direct User Access to Linear Frame Buffer.
10171 <1> ; Setup user's page tables for direct access to LFB.
10172 <1> ;
10173 <1> ; Permission checks are not implemented yet !
10174 <1> ; (10/12/2020)
10175 <1>
10176 0000F5EE 80FBFF <1> cmp bl, 0FFh ; current video mode
10177 0000F5F1 722C <1> jb short sysvideo_17_2 ; for desired video mode
10178 <1>
10179 0000F5F3 381D[1E680000] <1> cmp [CRT_MODE], bl ; VESA VBE video mode ?
10180 0000F5F9 750E <1> jne short sysvideo_17_1
10181 0000F5FB 668B0D[0E9D0100] <1> mov cx, [video_mode]
10182 0000F602 6681E1FF01 <1> and cx, 1FFh
10183 0000F607 EB29 <1> jmp short sysvideo_17_3
10184 <1> sysvideo_17_1:
10185 <1> ; 11/12/2020
10186 0000F609 88DF <1> mov bh, bl ; 0FFh
10187 0000F60B 8A1D[1E680000] <1> mov bl, [CRT_MODE] ; VGA/CGA video mode
10188 0000F611 8B2D[508E0100] <1> mov ebp, [u.usp] ; ebp points to user's registers
10189 <1> ; 23/12/2020
10190 0000F617 895D10 <1> mov [ebp+16], ebx ; return to user with EBX value
10191 0000F61A E95FD4FFFF <1> jmp sysret ; return to user with EAX = 0
10192 <1> sysvideo_17_2:
10193 <1> ; bl = VESA video mode - 100h
10194 0000F61F B701 <1> mov bh, 1 ; bx = 1xxh
10195 0000F621 53 <1> push ebx ; requested vesa video mode
10196 0000F622 E88C44FFFF <1> call vbe_biosfn_return_current_mode
10197 0000F627 59 <1> pop ecx ; requested vesa video mode
10198 0000F628 6681E3FF01 <1> and bx, 1FFh
10199 0000F62D 6639D9 <1> cmp cx, bx
10200 0000F630 7564 <1> jne short sysvideo_17_8
10201 <1> sysvideo_17_3:
10202 0000F632 663B0D[1A9D0100] <1> cmp cx, [LFB_Info+LFBINFO.mode]
10203 0000F639 755B <1> jne short sysvideo_17_8
10204 <1> sysvideo_17_4:
10205 <1> ; 11/12/2020
10206 0000F63B A1[1C9D0100] <1> mov eax, [LFB_Info+LFBINFO.LFB_addr]
10207 <1> ; 21/12/2020
10208 0000F640 09C0 <1> or eax, eax
10209 0000F642 744D <1> jz short sysvideo_17_7
10210 <1> ;
10211 0000F644 8B0D[209D0100] <1> mov ecx, [LFB_Info+LFBINFO.LFB_size] ; buff size
10212 0000F64A 89C3 <1> mov ebx, eax ; user's address = physical address
10213 <1> ;push ebx
10214 0000F64C 51 <1> push ecx
10215 <1> ; 21/12/2020
10216 0000F64D 81C1FF0F0000 <1> add ecx, 4095 ; PAGESIZE - 1
10217 <1> ; 14/12/2020
10218 0000F653 C1E90C <1> shr ecx, 12 ; convert bytes to pages
10219 0000F656 E86869FFFF <1> call direct_memory_access
10220 0000F65B 5A <1> pop edx ; linear frame buffer size in bytes
10221 <1> ;pop eax ; linear frame buffer address (physical)
10222 0000F65C 7233 <1> jc short sysvideo_17_7 ; [u.r0] = eax = 0
10223 <1> sysvideo_17_5:
10224 0000F65E 668B0D[269D0100] <1> mov cx, [LFB_Info+LFBINFO.Y_res] ; screen height
10225 0000F665 C1E110 <1> shl ecx, 16
10226 0000F668 668B0D[249D0100] <1> mov cx, [LFB_Info+LFBINFO.X_res] ; screen width
10227 0000F66F 31DB <1> xor ebx, ebx
10228 0000F671 8A1D[289D0100] <1> mov bl, [LFB_Info+LFBINFO.bpp] ; bits per pixel
10229 0000F677 8A3D[1A9D0100] <1> mov bh, [LFB_Info+LFBINFO.mode] ; XX part of 1xxh
10230 <1> sysvideo_26_4: ; 23/12/2020
10231 0000F67D 8B2D[508E0100] <1> mov ebp, [u.usp] ; ebp points to user's registers
10232 0000F683 895514 <1> mov [ebp+20], edx ; return to user with EDX value
10233 0000F686 895D10 <1> mov [ebp+16], ebx ; EBX
10234 0000F689 894D18 <1> mov [ebp+24], ecx ; ECX
10235 <1> sysvideo_17_6:
10236 0000F68C A3[548E0100] <1> mov [u.r0], eax ; LFB address
10237 <1> sysvideo_17_7:
10238 0000F691 E9E8D3FFFF <1> jmp sysret
10239 <1> sysvideo_17_8:
10240 <1> ; cx = mode
10241 <1> ; 21/12/2020
10242 0000F696 80CD40 <1> or ch, 40h ; Linear frame buffer flag
10243 0000F699 E83842FFFF <1> call _vbe_biosfn_return_mode_info
10244 0000F69E 72F1 <1> jc short sysvideo_17_7
10245 0000F6A0 EB99 <1> jmp short sysvideo_17_4
10246 <1>
10247 <1> sysvideo_19:
10248 <1> ; 23/07/2022
10249 <1> ; 22/01/2021
10250 <1> ; 12/12/2020
10251 <1> ; 11/12/2020
10252 <1> ; 23/11/2020
10253 <1> ; BH = 7
10254 <1> ; Get (Super/Extended VGA) mode
10255 <1> ; and Linear Frame Buffer info.

```

```

10256 <1>
10257 <1> ; 22/01/2021
10258 0000F6A2 B3FF <1> mov bl, 0FFh
10259 <1> ; 11/12/2020
10260 <1> ; cmp byte [CRT_MODE], 0FFh ; (extended mode?)
10261 <1> ; 22/01/2021
10262 0000F6A4 381D[1E680000] <1> cmp [CRT_MODE], bl ; 0FFh
10263 <1> ; jnb short sysvideo_17_1 ; not a VESA VBE mode
10264 <1> ; 23/07/2022
10265 <1> ; 12/12/2020
10266 0000F6AA 7305 <1> jnb short sysvideo_19_0
10267 <1> ; 23/07/2022
10268 0000F6AC E958FFFFFF <1> jmp sysvideo_17_1
10269 <1>
10270 <1> sysvideo_19_0:
10271 0000F6B1 E8FD43FFFF <1> call vbe_biosfn_return_current_mode
10272 0000F6B6 6681E3FF01 <1> and bx, 1FFh
10273 0000F6BB 663B1D[1A9D0100] <1> cmp bx, [LFB_Info+LFBINFO.mode]
10274 0000F6C2 750D <1> jne short sysvideo_19_2
10275 <1> sysvideo_19_1:
10276 0000F6C4 A1[1C9D0100] <1> mov eax, [LFB_Info+LFBINFO.LFB_addr]
10277 0000F6C9 8B15[209D0100] <1> mov edx, [LFB_Info+LFBINFO.LFB_size]
10278 0000F6CF EB8D <1> jmp sysvideo_17_5
10279 <1> sysvideo_19_2:
10280 0000F6D1 E80042FFFF <1> call _vbe_biosfn_return_mode_info
10281 0000F6D6 73EC <1> jnc short sysvideo_19_1
10282 0000F6D8 E9A1D3FFFF <1> jmp sysret
10283 <1>
10284 <1> sysvideo_20_1:
10285 <1> ; cx = vesa video mode
10286 0000F6DD 6689C8 <1> mov ax, cx
10287 0000F6E0 663D0001 <1> cmp ax, 100h
10288 0000F6E4 725C <1> jb short sysvideo_20_0 ; VGA/CGA mode
10289 0000F6E6 663DFF01 <1> cmp ax, 1FFh
10290 <1> ; ja short sysvideo_20_4 ; not valid
10291 0000F6EA 773E <1> ja short sysvideo_20_3
10292 0000F6EC 50 <1> push eax
10293 0000F6ED 6689C3 <1> mov bx, ax
10294 0000F6F0 66B8024F <1> mov ax, 4F02h
10295 <1>
10296 <1> ; simulate _int10h (int 31h) for func 4F02h
10297 <1> ; pushfd
10298 <1> ; push cs
10299 <1> ; push sysvideo_20_1_retn
10300 <1> ; push es ; *
10301 <1> ; push ds ; ** ; SAVE WORK AND PARAMETER REGISTERS
10302 <1> ; jmp VBE_func
10303 <1> ; sysvideo_20_1_retn:
10304 <1>
10305 0000F6F4 E82B20FFFF <1> call _int10h ; simulate int 10h (int 31h)
10306 <1>
10307 0000F6F9 6683F84F <1> cmp ax, 004Fh
10308 0000F6FD 58 <1> pop eax
10309 0000F6FE 752A <1> jne short sysvideo_20_3 ; error
10310 <1> ; pop eax
10311 0000F700 40 <1> inc eax
10312 0000F701 A3[548E0100] <1> mov [u.r0], eax ; video mode + 1
10313 0000F706 09D2 <1> or edx, edx ; is LFBINFO requested by user ?
10314 <1> ; jz short sysvideo_20_4
10315 0000F708 7420 <1> jz short sysvideo_20_3 ; no
10316 <1>
10317 <1> ; 11/12/2020
10318 <1> ; Check LFBINFO table/structure
10319 <1> ; (it is set by vbe2 'vbe_biosfn_set_mode'
10320 <1> ; but if vbe3 vbios pmi is in use,
10321 <1> ; it will not set LFBINFO table)
10322 <1>
10323 0000F70A 52 <1> push edx
10324 0000F70B 48 <1> dec eax ; video mode
10325 0000F70C BE[1A9D0100] <1> mov esi, LFB_Info
10326 0000F711 663B06 <1> cmp ax, [esi+LFBINFO.mode]
10327 0000F714 7407 <1> je short sysvideo_20_2
10328 <1>
10329 0000F716 E8BB41FFFF <1> call _vbe_biosfn_return_mode_info
10330 <1> ; jnc short sysvideo_20_2
10331 0000F71B 723B <1> jc short sysvideo_20_4 ; edx = 0
10332 <1>
10333 <1> ; ; clear LFBINFO table for invalidating
10334 <1> ; mov ecx, LFBINFO.size ; 16
10335 <1> ; mov edi, esi ; LFB_Info table address
10336 <1> ; xor eax, eax
10337 <1> ; rep stosb
10338 <1>
10339 <1> sysvideo_20_2:
10340 <1> ; pop ecx
10341 <1> ; mov edi, ecx ; user buffer
10342 0000F71D 5F <1> pop edi
10343 0000F71E B910000000 <1> mov ecx, LFBINFO.size ; 16
10344 0000F723 E8E3130000 <1> call transfer_to_user_buffer ; fast transfer
10345 0000F728 722F <1> jc short sysvideo_20_5
10346 <1>
10347 <1> ; jmp sysret
10348 <1> sysvideo_20_3:
10349 <1> ; pop eax ; [u.r0] = 0
10350 <1> ; sysvideo_20_4:
10351 0000F72A E94FD3FFFF <1> jmp sysret
10352 <1>
10353 <1> ; 23/07/2022
10354 <1> sysvideo_20:
10355 <1> ; 11/12/2020
10356 <1> ; 23/11/2020
10357 0000F72F 80FF08 <1> cmp bh, 8
10358 <1> ; 08/08/2022
10359 <1> ; jnb short sysvideo_19 ; video mode & lfb info
10360 0000F732 7407 <1> je short sysvideo_20_6
10361 <1> ; 23/07/2022
10362 0000F734 7733 <1> ja short sysvideo_21 ; 12/12/2020
10363 <1> ; 08/08/2022 - jnb
10364 0000F736 E967FFFFFF <1> jmp sysvideo_19
10365 <1> sysvideo_20_6:
10366 <1> ; BH = 8
10367 <1> ; Set (Super/Extended VGA) mode & return LFB info
10368 <1>
10369 <1> ; 11/12/2020
10370 0000F73B 80FBFF <1> cmp bl, 0FFh ; CGA/VGA mode ?
10371 0000F73E 739D <1> jnb short sysvideo_20_1
10372 <1>
10373 <1> ; xor ah, ah
10374 0000F740 88D8 <1> mov al, bl
10375 <1> sysvideo_20_0:
10376 0000F742 E8DD1FFFFFFF <1> call _int10h ; uses vbe3 pmi32 option
10377 0000F747 83F8FF <1> cmp eax, 0FFFFFFFh ; -1
10378 0000F74A 74DE <1> je short sysvideo_20_3 ; error
10379 <1>

```

```

10380 <1> ; 11/12/2020
10381 <1> ; alternative (it does not use vbe3 pmi32)
10382 <1> ;push eax
10383 <1> ;call _set_mode
10384 <1> ;pop eax
10385 <1> ;jc short sysvideo_20_3
10386 <1>
10387 <1> ;inc eax
10388 0000F74C FEC0 <1> inc al
10389 <1> ;mov [u.r0], ax ; video mode + 1
10390 0000F74E A2[548E0100] <1> mov [u.r0], al
10391 0000F753 E926D3FFFF <1> jmp sysret
10392 <1>
10393 <1> sysvideo_20_4:
10394 0000F758 5A <1> pop edx
10395 <1> sysvideo_20_5:
10396 0000F759 31D2 <1> xor edx, edx ; 0
10397 <1> ; edx = 0 -> invalid LFBINFO data
10398 0000F75B 8B2D[508E0100] <1> mov ebp, [u.usp] ; ebp points to user's registers
10399 0000F761 895514 <1> mov [ebp+20], edx ; return to user with EDX value
10400 0000F764 E915D3FFFF <1> jmp sysret
10401 <1>
10402 <1> sysvideo_21:
10403 <1> ; 23/07/2022
10404 <1> ; 04/01/2021
10405 <1> ; 03/12/2020
10406 0000F769 80FF0A <1> cmp bh, 10
10407 <1> ;jb sysvideo_22 ; VESA VBE3 pmi parms
10408 <1> ; 23/07/2022
10409 0000F76C 723F <1> jb short sysvideo_21_19
10410 <1> ; 23/12/2020
10411 <1> ;je sysvideo_26 ; video memory mapping
10412 <1> ; 23/07/2022
10413 0000F76E 7442 <1> je short sysvideo_21_20
10414 <1>
10415 <1> ; 04/01/2020
10416 0000F770 80FF0B <1> cmp bh, 11
10417 <1> ;ja sysvideo_27
10418 <1> ; 23/07/2022
10419 0000F773 7742 <1> ja short sysvideo_21_21
10420 <1>
10421 <1> ; BH = 11
10422 <1> ; set/read DAC color registers (for 8bpp)
10423 <1>
10424 0000F775 80FB04 <1> cmp bl, 4
10425 <1> ;jnb sysvideo_21_7 ; BMP file type palette
10426 <1> ; handling
10427 <1> ; 23/07/2022
10428 <1> ;jnb short sysvideo_21_7
10429 <1> ; 08/08/2022
10430 0000F778 7205 <1> jb short sysvideo_21_18
10431 0000F77A E989000000 <1> jmp sysvideo_21_7
10432 <1> sysvideo_21_18:
10433 0000F77F F6C301 <1> test bl, 1
10434 0000F782 7563 <1> jnz short sysvideo_21_4 ; set/write DAC colors
10435 <1>
10436 <1> ; Read DAC color register or all DAC color registers
10437 0000F784 F6C302 <1> test bl, 2 ; read single DAC color register
10438 0000F787 7433 <1> jz short sysvideo_21_2 ; read all DAC color regs
10439 <1>
10440 <1> ; read single DAC color register
10441 <1> ; CL = DAC color register (index)
10442 <1>
10443 0000F789 66BAC703 <1> mov dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
10444 0000F78D 88C8 <1> mov al, cl ; DAC color register
10445 0000F78F 31C9 <1> xor ecx, ecx ; (this may not be necessary)
10446 0000F791 EE <1> out dx, al
10447 <1> ;mov dx, 3C9h ; VGAREG_DAC_DATA
10448 0000F792 B2C9 <1> mov dl, 0C9h
10449 0000F794 EC <1> in al, dx
10450 0000F795 88C4 <1> mov ah, al ; red
10451 0000F797 EC <1> in al, dx
10452 0000F798 88C1 <1> mov cl, al ; green
10453 0000F79A EC <1> in al, dx
10454 0000F79B 88C5 <1> mov ch, al ; blue
10455 0000F79D C1E108 <1> shl ecx, 8
10456 0000F7A0 88E1 <1> mov cl, ah ; red
10457 <1> ; CL = Red, CH = Green, byte 3 = Blue, byte 4 = 0
10458 <1> sysvideo_21_0:
10459 0000F7A2 890D[548E0100] <1> mov [u.r0], ecx
10460 <1> sysvideo_21_1:
10461 0000F7A8 E9D1D2FFFF <1> jmp sysret
10462 <1> sysvideo_21_19:
10463 <1> ; 23/07/2022
10464 0000F7AD E97A010000 <1> jmp sysvideo_22
10465 <1> sysvideo_21_20:
10466 <1> ; 23/07/2022
10467 0000F7B2 E930020000 <1> jmp sysvideo_26
10468 <1> sysvideo_21_21:
10469 <1> ; 23/07/2022
10470 0000F7B7 E9B6020000 <1> jmp sysvideo_27
10471 <1> sysvideo_21_2:
10472 <1> ; read all DAC color registers
10473 0000F7BC 89CB <1> mov ebx, ecx ; user's buffer address
10474 0000F7BE BF00600900 <1> mov edi, VBE3STACKADDR
10475 0000F7C3 89FE <1> mov esi, edi
10476 <1> ;mov ecx, 768 ; 256*3
10477 <1> ; 08/08/2022
10478 0000F7C5 29C9 <1> sub ecx, ecx
10479 0000F7C7 B503 <1> mov ch, 3
10480 <1> ; ecx = 768 = 300h
10481 <1> ;push ecx
10482 0000F7C9 66BAC703 <1> mov dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
10483 0000F7CD 28C0 <1> sub al, al ; 0
10484 0000F7CF EE <1> out dx, al
10485 <1> ;mov dx, 3C9h ; VGAREG_DAC_DATA
10486 0000F7D0 B2C9 <1> mov dl, 0C9h
10487 <1> sysvideo_21_3:
10488 0000F7D2 EC <1> in al, dx
10489 0000F7D3 AA <1> stosb
10490 0000F7D4 EC <1> in al, dx
10491 0000F7D5 AA <1> stosb
10492 0000F7D6 EC <1> in al, dx
10493 0000F7D7 AA <1> stosb
10494 0000F7D8 E2F8 <1> loop sysvideo_21_3
10495 <1> ;pop ecx
10496 <1> ; 18/08/2022
10497 0000F7DA B503 <1> mov ch, 3
10498 <1> ; ecx = 768 = 300h
10499 <1>
10500 0000F7DC 89DF <1> mov edi, ebx ; user's buffer address
10501 <1> ;mov esi, VBE3STACKADDR
10502 <1> ;mov ecx, 256*3 = 768
10503 0000F7DE E828130000 <1> call transfer_to_user_buffer

```

```

10504 0000F7E3 72C3      <1>      jc      short sysvideo_21_1
10505                        <1>      ;mov     [u.r0], ecx ; actual transfer count
10506 0000F7E5 EBBB      <1>      jmp     short sysvideo_21_0
10507                        <1>
10508                        <1> sysvideo_21_4:
10509                        <1>      ; Set/write DAC color register or all registers
10510 0000F7E7 F6C302     <1>      test    bl, 2 ; write/set single DAC color register
10511 0000F7EA 7456      <1>      jz      short sysvideo_21_5 ; set all DAC color regs
10512                        <1>
10513                        <1>      ; set single DAC color register
10514                        <1>      ; CL = DAC color register (index)
10515                        <1>      ; (byte 1 = Red, byte 2 = Green, byte 3 = Blue)
10516                        <1>
10517 0000F7EC 66BAC803   <1>      mov     dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
10518 0000F7F0 89C8      <1>      mov     eax, ecx ; DAC color register (index)
10519 0000F7F2 C1E910     <1>      shr     ecx, 16 ; CL = green, AH = Red
10520 0000F7F5 EE        <1>      out     dx, al
10521                        <1>      ;mov     dx, 3C9h ; VGAREG_DAC_DATA
10522 0000F7F6 FEC2      <1>      inc     dl
10523 0000F7F8 88E0      <1>      mov     al, ah ; Red
10524 0000F7FA EE        <1>      out     dx, al
10525 0000F7FB 88C8      <1>      mov     al, cl ; Green
10526 0000F7FD EE        <1>      out     dx, al
10527 0000F7FE 88E8      <1>      mov     al, ch ; Blue
10528 0000F800 EE        <1>      out     dx, al
10529                        <1>      ;rol     ecx, 8
10530 0000F801 C1E108     <1>      shl     ecx, 8 ; 21/02/2021
10531 0000F804 88E1      <1>      mov     cl, ah ; Red
10532                        <1>      ; ecx = 00BBGGRRh
10533 0000F806 EB9A      <1>      jmp     short sysvideo_21_0
10534                        <1>
10535                        <1>      ; 23/07/2022
10536                        <1> sysvideo_21_7:
10537                        <1>      ; BMP file type palette handling
10538                        <1>
10539 0000F808 F6C301     <1>      test    bl, 1
10540                        <1>      ;jnz     short sysvideo_21_12 ; set/write DAC colors
10541                        <1>      ; 08/08/2022
10542 0000F80B 7405      <1>      jz      short sysvideo_21_16
10543 0000F80D E9A3000000 <1>      jmp     sysvideo_21_12
10544                        <1>
10545                        <1> sysvideo_21_16:
10546                        <1>      ; Read DAC color register or all DAC color registers
10547 0000F812 F6C302     <1>      test    bl, 2 ; read single DAC color register
10548 0000F815 7460      <1>      jz      short sysvideo_21_10 ; read all DAC color regs
10549                        <1>
10550                        <1>      ; read single DAC color register
10551                        <1>      ; CL = DAC color register (index)
10552                        <1>
10553 0000F817 66BAC703   <1>      mov     dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
10554 0000F81B 88C8      <1>      mov     al, cl ; DAC color register
10555 0000F81D 31C9      <1>      xor     ecx, ecx
10556 0000F81F EE        <1>      out     dx, al
10557                        <1>      ;mov     dx, 3C9h ; VGAREG_DAC_DATA
10558 0000F820 B2C9      <1>      mov     dl, 0C9h
10559 0000F822 EC        <1>      in     al, dx
10560 0000F823 C0E002     <1>      shl     al, 2
10561 0000F826 88C5      <1>      mov     ch, al ; red
10562 0000F828 EC        <1>      in     al, dx
10563 0000F829 C0E002     <1>      shl     al, 2
10564 0000F82C 88C1      <1>      mov     cl, al ; green
10565 0000F82E EC        <1>      in     al, dx
10566 0000F82F C0E002     <1>      shl     al, 2
10567                        <1>      ; 21/02/2021
10568 0000F832 C1E108     <1>      shl     ecx, 8
10569 0000F835 88C1      <1>      mov     cl, al ; blue
10570                        <1>      ; CL = Blue, CH = Green, byte 3 = Red, byte 4 = 0
10571                        <1> sysvideo_21_8:
10572 0000F837 890D[548E0100] <1>      mov     [u.r0], ecx
10573                        <1> sysvideo_21_9:
10574 0000F83D E93CD2FFFF <1>      jmp     sysret
10575                        <1>
10576                        <1> sysvideo_21_5:
10577                        <1>      ; write/set all DAC color registers
10578 0000F842 89CE      <1>      mov     esi, ecx ; user's buffer address
10579 0000F844 BF00600900 <1>      mov     edi, VBE3STACKADDR
10580 0000F849 89FB      <1>      mov     ebx, edi
10581 0000F84B B900030000 <1>      mov     ecx, 768 ; 256*3
10582 0000F850 E800130000 <1>      call    transfer_from_user_buffer
10583                        <1>      ;jc      short sysvideo_21_1
10584                        <1>      ; 08/08/2022
10585 0000F855 7305      <1>      jnc     short sysvideo_21_17
10586 0000F857 E94CFFFFFF <1>      jmp     sysvideo_21_1
10587                        <1> sysvideo_21_17:
10588 0000F85C 890D[548E0100] <1>      mov     [u.r0], ecx ; actual transfer count
10589                        <1>
10590 0000F862 89DE      <1>      mov     esi, ebx ; VBE3STACKADDR
10591 0000F864 66BAC803   <1>      mov     dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
10592 0000F868 28C0      <1>      sub     al, al ; 0
10593 0000F86A EE        <1>      out     dx, al
10594                        <1>      ;mov     dx, 3C9h ; VGAREG_DAC_DATA
10595 0000F86B FEC2      <1>      inc     dl
10596                        <1> sysvideo_21_6:
10597 0000F86D AC        <1>      lodsb
10598 0000F86E EE        <1>      out     dx, al
10599 0000F86F AC        <1>      lodsb
10600 0000F870 EE        <1>      out     dx, al
10601 0000F871 AC        <1>      lodsb
10602 0000F872 EE        <1>      out     dx, al
10603 0000F873 E2F8      <1>      loop    sysvideo_21_6
10604 0000F875 EBC6      <1>      jmp     short sysvideo_21_9
10605                        <1>
10606                        <1> sysvideo_21_10:
10607                        <1>      ; read all DAC color registers
10608 0000F877 89CD      <1>      mov     ebp, ecx ; user's buffer address
10609 0000F879 BF00600900 <1>      mov     edi, VBE3STACKADDR
10610 0000F87E 89FE      <1>      mov     esi, edi
10611 0000F880 B900040000 <1>      mov     ecx, 1024 ; 256*4
10612 0000F885 51        <1>      push    ecx
10613 0000F886 66BAC703   <1>      mov     dx, 3C7h ; VGAREG_DAC_READ_ADDRESS
10614 0000F88A 28C0      <1>      sub     al, al ; 0
10615 0000F88C EE        <1>      out     dx, al
10616                        <1>      ;mov     dx, 3C9h ; VGAREG_DAC_DATA
10617 0000F88D B2C9      <1>      mov     dl, 0C9h
10618                        <1> sysvideo_21_11:
10619 0000F88F 31DB      <1>      xor     ebx, ebx
10620 0000F891 EC        <1>      in     al, dx ; Red
10621 0000F892 C0E002     <1>      shl     al, 2
10622 0000F895 88C7      <1>      mov     bh, al
10623 0000F897 EC        <1>      in     al, dx ; Green
10624 0000F898 C0E002     <1>      shl     al, 2
10625 0000F89B 88C3      <1>      mov     bl, al
10626 0000F89D EC        <1>      in     al, dx ; Blue
10627 0000F89E C0E002     <1>      shl     al, 2

```

```

10628 0000F8A1 C1E308 <1> shl ebx, 8
10629 0000F8A4 89D8 <1> mov eax, ebx ; 00RRGGBBh
10630 0000F8A6 AB <1> stosd
10631 0000F8A7 E2E6 <1> loop sysvideo_21_11
10632 0000F8A9 59 <1> pop ecx
10633 <1>
10634 0000F8AA 89EF <1> mov edi, ebp ; user's buffer address
10635 <1> ;mov esi, VBE3STACKADDR
10636 <1> ;mov ecx, 1024 = 4*256
10637 0000F8AC E85A120000 <1> call transfer_to_user_buffer
10638 0000F8B1 728A <1> jc short sysvideo_21_9
10639 <1> ;mov [u.r0], ecx ; actual transfer count
10640 0000F8B3 EB82 <1> jmp short sysvideo_21_8
10641 <1>
10642 <1> sysvideo_21_12:
10643 <1> ; Set/write DAC color register or all registers
10644 0000F8B5 F6C302 <1> test bl, 2 ; write/set single DAC color register
10645 0000F8B8 742A <1> jz short sysvideo_21_13 ; set all DAC color regs
10646 <1>
10647 <1> ; set single DAC color register
10648 <1> ; CL = DAC color register (index)
10649 <1> ; (byte 1 = Blue, byte 2 = Green, byte 3 = Red)
10650 <1>
10651 0000F8BA 66BAC803 <1> mov dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
10652 0000F8BE 88C8 <1> mov al, cl ; DAC color register (index)
10653 0000F8C0 88EC <1> mov ah, ch ; Blue
10654 0000F8C2 C1E910 <1> shr ecx, 16
10655 0000F8C5 EE <1> out dx, al
10656 <1> ;mov dx, 3C9h ; VGAREG_DAC_DATA
10657 0000F8C6 FEC2 <1> inc dl
10658 0000F8C8 88E8 <1> mov al, ch ; Red
10659 0000F8CA C0E802 <1> shr al, 2
10660 0000F8CD EE <1> out dx, al
10661 0000F8CE 88C8 <1> mov al, cl ; Green
10662 0000F8D0 C0E802 <1> shr al, 2
10663 0000F8D3 EE <1> out dx, al
10664 0000F8D4 88E0 <1> mov al, ah ; Blue
10665 0000F8D6 C0E802 <1> shr al, 2
10666 0000F8D9 EE <1> out dx, al
10667 <1> ;rol ecx, 8
10668 0000F8DA C1E108 <1> shl ecx, 8 ; 21/02/2021
10669 0000F8DD 88E1 <1> mov cl, ah
10670 0000F8DF E953FFFFFF <1> jmp sysvideo_21_8 ; 08/08/2022
10671 <1>
10672 <1> sysvideo_21_13:
10673 <1> ; write/set all DAC color registers
10674 0000F8E4 89CE <1> mov esi, ecx ; user's buffer address
10675 0000F8E6 BF00600900 <1> mov edi, VBE3STACKADDR
10676 0000F8EB 89FB <1> mov ebx, edi
10677 0000F8ED B900040000 <1> mov ecx, 1024 ; 256*4
10678 0000F8F2 E85E120000 <1> call transfer_from_user_buffer
10679 0000F8F7 722E <1> jc short sysvideo_21_15
10680 0000F8F9 890D[548E0100] <1> mov [u.r0], ecx ; actual transfer count
10681 <1>
10682 0000F8FF 89DE <1> mov esi, ebx ; VBE3STACKADDR
10683 0000F901 66BAC803 <1> mov dx, 3C8h ; VGAREG_DAC_WRITE_ADDRESS
10684 0000F905 28C0 <1> sub al, al ; 0
10685 0000F907 EE <1> out dx, al
10686 <1> ;mov dx, 3C9h ; VGAREG_DAC_DATA
10687 0000F908 FEC2 <1> inc dl
10688 <1> sysvideo_21_14:
10689 0000F90A AD <1> lodsd
10690 <1> ; byte 0 = Blue, byte 1 = Green, byte 2 = Red
10691 <1> ; 21/02/2021
10692 0000F90B 89C3 <1> mov ebx, eax ; BL = Blue, BH = Green
10693 0000F90D C1CB08 <1> ror ebx, 8 ; BL = Green, BH = Red
10694 0000F910 88F8 <1> mov al, bh
10695 0000F912 C0E802 <1> shr al, 2
10696 0000F915 EE <1> out dx, al ; Red
10697 0000F916 88D8 <1> mov al, bl
10698 0000F918 C0E802 <1> shr al, 2
10699 0000F91B EE <1> out dx, al ; Green
10700 0000F91C C1C308 <1> rol ebx, 8 ; BL = Blue
10701 0000F91F 88D8 <1> mov al, bl
10702 0000F921 C0E802 <1> shr al, 2
10703 0000F924 EE <1> out dx, al ; Blue
10704 0000F925 E2E3 <1> loop sysvideo_21_14
10705 <1> sysvideo_21_15:
10706 0000F927 E952D1FFFF <1> jmp sysret
10707 <1>
10708 <1> sysvideo_22:
10709 <1> ; 28/02/2021
10710 <1> ; 22/01/2021
10711 <1> ; 17/01/2021
10712 <1> ; 04/12/2020
10713 <1> ; 03/12/2020
10714 <1> ; BH = 9
10715 <1> ; Set/Get VESA VBE3 protected mode interface params
10716 <1>
10717 <1> ; 22/01/2021
10718 <1> ;cmp byte [vbe3], 3
10719 <1> ;jne short sysvideo_25 ; not applicable if
10720 <1> ; ; vbe3 compatible video bios
10721 <1> ; ; is not detected by kernel
10722 0000F92C 80FB02 <1> cmp bl, 2
10723 <1> ;ja short sysvideo_25 ; bl > 2 not implemented
10724 <1> ; 17/01/2021
10725 0000F92F 7716 <1> ja short sysvideo_22_0 ; srvs flag sub function
10726 <1> ;jb short sysvideo_23
10727 <1>
10728 <1> ; 21/01/2021
10729 0000F931 803D[86090000]03 <1> cmp byte [vbe3], 3
10730 <1> ;jne short sysvideo_25 ; not applicable if
10731 <1> ; ; vbe3 compatible video bios
10732 <1> ; ; is not detected by kernel
10733 0000F938 75ED <1> jne short sysvideo_21_15 ; 28/02/2021
10734 <1>
10735 0000F93A 80FB01 <1> cmp bl, 1
10736 0000F93D 7673 <1> jna short sysvideo_23
10737 <1>
10738 0000F93F 8A1D[0C9D0100] <1> mov bl, [pmi32] ; video bios 32 bit PMI functions
10739 0000F945 EB78 <1> jmp short sysvideo_24
10740 <1>
10741 <1> sysvideo_22_0:
10742 <1> ; 17/01/2021
10743 <1> ; save/restore video state user permission
10744 0000F947 80FB05 <1> cmp bl, 5
10745 0000F94A 771E <1> ja short sysvideo_22_2
10746 0000F94C 7208 <1> jb short sysvideo_22_1
10747 <1> ; get srvs flag value/status
10748 0000F94E 8A1D[709D0100] <1> mov bl, [srvsf] ; 0 = disabled, 1 = enabled
10749 0000F954 EB2C <1> jmp short sysvideo_22_3
10750 <1>
10751 <1> sysvideo_22_1:

```

```

10752      <1>      ; permission (root and multi tasking) check
10753 0000F956 E836000000 <1>      call sysvideo_22_4
10754 0000F95B 736A <1>      jnc short sysvideo_25 ; not permitted !
10755      <1>      ; cf = 1
10756 0000F95D 80EB03 <1>      sub b1, 3 ; disable = 0, enable = 1
10757      <1>      ; 22/01/2021
10758 0000F960 881D[709D0100] <1>      mov [srvsf], b1
10759 0000F966 FEC3 <1>      inc b1 ; 1 = disabled, 2 = enabled
10760 0000F968 EB18 <1>      jmp short sysvideo_22_3
10761      <1>
10762      <1> sysvideo_22_2:
10763 0000F96A 80FB06 <1>      cmp b1, 6
10764      <1>      ;ja short sysvideo_25 ; invalid/unimplemented
10765      <1>      ; 28/02/2021
10766 0000F96D 7733 <1>      ja short sysvideo_22_6
10767      <1>      ; get VESA VBE number/status
10768 0000F96F 8A25[86090000] <1>      mov ah, [vbe3] ; vbe3 = 3, vbe2 = 2, others = 0
10769 0000F975 A0[87090000] <1>      mov al, [vbe2bios] ; bochs/qemu/vbox emulator status
10770 0000F97A 66A3[548E0100] <1>      mov [u.r0], ax
10771 0000F980 EB45 <1>      jmp short sysvideo_25
10772      <1>
10773      <1> sysvideo_22_3:
10774      <1>      ; 22/01/2021
10775 0000F982 8A3D[719D0100] <1>      mov bh, [srvso] ; state options (> 80h -> svga)
10776 0000F988 66891D[548E0100] <1>      mov [u.r0], bx ; function result is return value
10777 0000F98F EB36 <1>      jmp short sysvideo_25
10778      <1>
10779      <1> sysvideo_22_4:
10780      <1>      ; 17/01/2021 - permission will be given by root only
10781 0000F991 803D[32830100]00 <1>      cmp byte [multi_tasking], 0 ; in single user mode
10782 0000F998 7707 <1>      ja short sysvideo_22_5
10783      <1>      ; 19/01/2021
10784 0000F99A 803D[A68E0100]01 <1>      cmp byte [u.uid], 1 ; ([u.uid] = 0 -> root)
10785      <1> sysvideo_22_5:
10786      <1>      ; [multi_tasking] = 0 & [u.uid] = 0 -> CF = 1
10787      <1>      ; otherwise -> CF = 0
10788 0000F9A1 C3 <1>      ret
10789      <1>
10790      <1> sysvideo_22_6:
10791      <1>      ; 28/02/2021
10792 0000F9A2 80FB09 <1>      cmp b1, 9
10793 0000F9A5 7720 <1>      ja short sysvideo_25 ; invalid/unimplemented
10794 0000F9A7 7436 <1>      je short sysvideo_22_9
10795 0000F9A9 80FB08 <1>      cmp b1, 8
10796 0000F9AC 721E <1>      jb short sysvideo_22_7
10797      <1>
10798      <1>      ; BL = 8
10799      <1>      ; Set default true color bpp to 24
10800      <1>
10801 0000F9AE B318 <1>      mov b1, 24
10802      <1>      ;mov [truecolor], al ; 24bpp (RRGGBBh)
10803      <1>      ;mov [u.r0], al
10804      <1>      ;jmp short sysvideo_25
10805 0000F9B0 EB25 <1>      jmp short sysvideo_22_8
10806      <1>
10807      <1> sysvideo_23:
10808      <1>      ; 17/01/2021
10809      <1>      ; permission (root and multi tasking) check
10810 0000F9B2 E8DAFFFFFF <1>      call sysvideo_22_4
10811 0000F9B7 730E <1>      jnc short sysvideo_25 ; not permitted !
10812      <1>
10813 0000F9B9 881D[0C9D0100] <1>      mov [pmi32], b1 ; 1 = enabled, 0 = disabled
10814      <1> sysvideo_24:
10815 0000F9BF FEC3 <1>      inc b1
10816      <1> sysvideo_22_10: ; 28/02/2021
10817 0000F9C1 881D[548E0100] <1>      mov [u.r0], b1 ; function result is return value
10818      <1> sysvideo_25:
10819 0000F9C7 E9B2D0FFFF <1>      jmp sysret
10820      <1>
10821      <1> sysvideo_22_7:
10822      <1>      ; BL = 7
10823      <1>      ; Set default true color bpp to 32
10824      <1>      ; (it will set if [VBE3]=3)
10825      <1>
10826      <1>      ; Note: This sub function is used to set 24bpp
10827      <1>      ; VESA VBE video modes to 32bpp.. because,
10828      <1>      ; old hardware uses 24 bpp but new video hardware
10829      <1>      ; uses 32bpp for same VESA VBE truecolor modes.
10830      <1>      ; (For example: VBE mode 112h is 640*480, 24bpp but
10831      <1>      ; new hardware uses/apply it as 640*480, 32bpp.)
10832      <1>      ; So, TRDOS 386 v2.0.3 kernel will check [truecolor]
10833      <1>      ; status is 32 bpp or not and it will change 24bpp
10834      <1>      ; to 32bpp if default [truecolor] value is 32, for
10835      <1>      ; same video mode number.
10836      <1>
10837 0000F9CC 803D[86090000]03 <1>      cmp byte [vbe3], 3
10838 0000F9D3 75F2 <1>      jne short sysvideo_25 ; Only applicable
10839      <1>      ; for VBE3 video hardware!
10840 0000F9D5 B320 <1>      mov b1, 32
10841      <1> sysvideo_22_8:
10842 0000F9D7 881D[63730100] <1>      mov [truecolor], b1 ; 32bpp (00RRGGBBh)
10843      <1>      ;mov [u.r0], b1
10844      <1>      ;jmp short sysvideo_25
10845 0000F9DD EBE2 <1>      jmp short sysvideo_22_10
10846      <1>
10847      <1> sysvideo_22_9:
10848      <1>      ; BL = 9
10849      <1>      ; Return default true color bpp
10850 0000F9DF 8A1D[63730100] <1>      mov b1, [truecolor]
10851 0000F9E5 EBDA <1>      jmp short sysvideo_22_10
10852      <1> ;sysvideo_22_10:
10853      <1>      ;mov [u.r0], b1
10854      <1>      ;jmp sysret
10855      <1>
10856      <1> sysvideo_26:
10857      <1>      ; 23/12/2020
10858      <1>      ; BH = 10
10859      <1>      ; Map video memory to user's buffer
10860      <1>      ; (multiuser/owner r/w permissions are ignored
10861      <1>      ; for current TRDOS 386 version !)
10862      <1>
10863 0000F9E7 6681E100F0 <1>      and cx, ~4095 ; clear low 12 bits
10864 0000F9EC 09C9 <1>      or ecx, ecx ; start address of user's buffer
10865 0000F9EE 74D7 <1>      jz short sysvideo_25 ; error !
10866      <1>
10867 0000F9F0 80FB01 <1>      cmp b1, 1 ; VGA memory mapping ?
10868 0000F9F3 740E <1>      je short sysvideo_26_1
10869 0000F9F5 7718 <1>      ja short sysvideo_26_2
10870      <1> sysvideo_26_0:
10871      <1>      ; BL = 0 : CGA memory (0B8000h) map (32K)
10872 0000F9F7 B800800B00 <1>      mov eax, 0B8000h
10873 0000F9FC B800800000 <1>      mov ebx, 32768
10874 0000FA01 EB37 <1>      jmp short sysvideo_26_3
10875      <1> sysvideo_26_1:

```

```

10876      <1>      ; BL = 1 : VGA memory (0A0000h) map (64K)
10877 0000FA03 B800000A00 <1>      mov     eax, 0A0000h
10878 0000FA08 BB00000100 <1>      mov     ebx, 65536
10879 0000FA0D EB2B <1>      jmp     short sysvideo_26_3
10880 <1>      sysvideo_26_2:
10881 <1>      ; BL = 2 : SVGA memory (LFB) map to user's buffer
10882 0000FA0F 803D[86090000]02 <1>      cmp     byte [vbe3], 2 ; VESA VBE 2/3 vbios ready ?
10883 0000FA16 72AF <1>      jnb     short sysvideo_25 ; no, error !
10884 0000FA18 6681E200F0 <1>      and     dx, ~4095 ; clear low 12 bits
10885 0000FA1D 09D2 <1>      or      edx, edx ; buffer size in bytes
10886 0000FA1F 74A6 <1>      jz      short sysvideo_25 ; error
10887 0000FA21 89D3 <1>      mov     ebx, edx
10888 0000FA23 A1[1C9D0100] <1>      mov     eax, [LFB_ADDR] ; [LFB_Info+LFBINFO.LFB_addr]
10889 0000FA28 21C0 <1>      and     eax, eax
10890 0000FA2A 7425 <1>      jz      short sysvideo_26_5
10891 <1>      ; (LFB parms are not set yet)
10892 0000FA2C 3B1D[209D0100] <1>      cmp     ebx, [LFB_SIZE] ; [LFB_Info+LFBINFO.LFB_size]
10893 0000FA32 7606 <1>      jna     short sysvideo_26_3
10894 0000FA34 8B1D[209D0100] <1>      mov     ebx, [LFB_SIZE]
10895 <1>      sysvideo_26_3:
10896 0000FA3A 52 <1>      push    edx
10897 0000FA3B 53 <1>      push    ebx ; buffer size in bytes
10898 0000FA3C 51 <1>      push    ecx ; user's buffer address
10899 0000FA3D 87CB <1>      xchg    ebx, ecx
10900 0000FA3F C1E90C <1>      shr     ecx, 12 ; convert buffer size to page count
10901 0000FA42 E87C65FFFF <1>      call    direct_memory_access
10902 0000FA47 59 <1>      pop     ecx ; user's buffer address
10903 0000FA48 5B <1>      pop     ebx ; buffer size
10904 0000FA49 5A <1>      pop     edx
10905 <1>      ;jc     short sysvideo_25 ; error !
10906 <1>      ; [u.r0] = 0
10907 <1>      ; 28/02/2021
10908 0000FA4A 7235 <1>      jc      short sysvideo_27_0 ; error !
10909 <1>
10910 <1>      ;sysvideo_26_4:
10911 <1>      ;mov     ebp, [u.usp] ; ebp points to user's registers
10912 <1>      ;mov     [ebp+20], edx ; return to user with EDX value
10913 <1>      ;mov     [ebp+16], ebx ; EBX
10914 <1>      ;mov     [ebp+24], ecx ; ECX
10915 <1>      ; eax = physical address of video memory (LFB)
10916 <1>      ;mov     [u.r0], eax
10917 <1>      ;jmp     sysret
10918 0000FA4C E92CFCFFFF <1>      jmp     sysvideo_26_4
10919 <1>
10920 <1>      sysvideo_26_5:
10921 0000FA51 66A1[1A0F0000] <1>      mov     ax, [def_LFB_addr] ; default LFB for mode 118h
10922 <1>      ; ah must be 0C0h or 0D0h or E0h
10923 <1>      ; others are nonsense !?
10924 0000FA57 08E4 <1>      or      ah, ah
10925 <1>      ;jz      short sysvideo_25 ; invalid lfb addr or
10926 <1>      ; it is not a vbe2.-bochs emu-
10927 <1>      ; or vbe3 -real- video bios
10928 <1>      ; 28/02/2021
10929 0000FA59 7426 <1>      jz      short sysvideo_27_0 ; invalid LFB address
10930 <1>
10931 0000FA5B 80FCF0 <1>      cmp     ah, 0F0h
10932 <1>      ;jnb     short sysvideo_25 ; nonsense !?
10933 <1>      ; 28/02/2021
10934 0000FA5E 7321 <1>      jnb     short sysvideo_27_0 ; nonsense !?
10935 <1>
10936 0000FA60 C1E010 <1>      shl     eax, 16
10937 <1>      ;jz      short sysvideo_25 ; eax = 0
10938 <1>
10939 0000FA63 81FB00907E00 <1>      cmp     ebx, 1920*1080*4 ; maximum value of possible
10940 <1>      ; buffer sizes
10941 0000FA69 76CF <1>      jna     short sysvideo_26_3 ; buffer size is proper
10942 <1>      ; resize buffer to fit 4GB limit
10943 0000FA6B BB00907E00 <1>      mov     ebx, 1920*1080*4
10944 0000FA70 EBC8 <1>      jmp     short sysvideo_26_3
10945 <1>
10946 <1>      sysvideo_27:
10947 <1>      ; 23/07/2022
10948 <1>      ; 16/02/2021
10949 <1>      ; 18/01/2021
10950 0000FA72 80FF0C <1>      cmp     bh, 12
10951 <1>      ;ja     sysvideo_28 ; 19/01/2021
10952 <1>      ; 23/07/2022
10953 0000FA75 7605 <1>      jna     short sysvideo_27_24
10954 0000FA77 E970010000 <1>      jmp     sysvideo_28
10955 <1>
10956 <1>      sysvideo_27_24: ; 23/07/2022
10957 <1>      ; BH = 12
10958 <1>      ; Font sub functions.
10959 <1>      ; 12/02/2021
10960 <1>      ; 11/01/2021
10961 <1>      ; 10/01/2021
10962 <1>      ; BL = 0 : Disable system font overwrite
10963 <1>      ; BL = 1 : Enable system font overwrite
10964 <1>      ; BL = 2 : Read system font 8x8
10965 <1>      ; BL = 3 : Read system font 8x14
10966 <1>      ; BL = 4 : Read system font 8x16
10967 <1>      ; BL = 5 : Read user defined font 8x8
10968 <1>      ; BL = 6 : Read user defined font 8x16
10969 <1>      ; BL = 7 : Write system font 8x8
10970 <1>      ; BL = 8 : Write system font 8x14
10971 <1>      ; BL = 9 : Write system font 8x16
10972 <1>      ; BL = 10 : Write user defined font 8x8
10973 <1>      ; BL = 11 : Write user defined font 8x16
10974 <1>      ;
10975 <1>      ; BL > 11 : invalid (not implemented)
10976 <1>      ;
10977 <1>      ; For BL = 1 to 11
10978 <1>      ; ECX = number of characters (<= 256)
10979 <1>      ; EDX = first character (ascii code in DL)
10980 <1>      ; ESI = user's buffer address
10981 <1>      ;
10982 <1>      ; Return: EAX = character count
10983 <1>
10984 0000FA7C 80FB0B <1>      cmp     bl, 11
10985 0000FA7F 7605 <1>      jna     short sysvideo_27_1
10986 <1>      sysvideo_27_0:
10987 0000FA81 E9F8CFFFFF <1>      jmp     sysret ; not implemented yet !
10988 <1>      sysvideo_27_1:
10989 0000FA86 66B80001 <1>      mov     ax, 256
10990 0000FA8A 08DB <1>      or      bl, bl
10991 0000FA8C 750E <1>      jnz     short sysvideo_27_3
10992 <1>
10993 <1>      ; bl = 0
10994 <1>      ; disable system font overwrite
10995 <1>
10996 0000FA8E 8025[6E9D0100]7F <1>      and     byte [ufont], 7Fh ; clear bit 7
10997 <1>      sysvideo_27_2:
10998 <1>      ;mov     word [u.r0], 256 ; > 0 -> successful
10999 0000FA95 A3[548E0100] <1>      mov     [u.r0], eax ; 256

```



```

11000 0000FA9A EBE5      <1>      jmp      short sysvideo_27_0
11001                               <1> sysvideo_27_3:
11002 0000FA9C 80FB01   <1>      cmp      bl, 1
11003 0000FA9F 7710     <1>      ja       short sysvideo_27_4
11004                               <1>
11005                               <1>      ; bl = 1
11006                               <1>      ; enable system font overwrite
11007                               <1>      ; if [multi_tasking]= 0 and [u.uid] = 0
11008                               <1>
11009                               <1>      ;cmp      byte [multi_tasking], 0
11010                               <1>      ;                               ; multi tasking enabled ?
11011                               <1>      ;ja       short sysvideo_27_0 ; yes
11012                               <1>      ;;; 19/01/2021
11013                               <1>      ;; system maintenance or single user mode
11014                               <1>      ;cmp      byte [u.uid], 0 ; root ?
11015                               <1>      ;ja       short sysvideo_27_0 ; no
11016                               <1>
11017                               <1>      ; 19/01/2021
11018                               <1>      ; multi tasking & root check
11019 0000FAA1 E8EBFEFFFF <1>      call     sysvideo_22_4
11020 0000FAA6 73D9     <1>      jnc      short sysvideo_27_0 ; not permitted
11021                               <1>
11022                               <1>      ; [multi_tasking]= 0 and [u.uid] = 0
11023                               <1>
11024 0000FAA8 800D[6E9D0100]80 <1>      or       byte [ufont], 80h ; set bit 7
11025                               <1>
11026 0000FAAF EBE4     <1>      jmp      short sysvideo_27_2
11027                               <1>
11028                               <1> sysvideo_27_4:
11029 0000FAB1 09C9     <1>      or       ecx, ecx
11030 0000FAB3 74CC     <1>      jz       short sysvideo_27_0
11031 0000FAB5 21D2     <1>      and      edx, edx
11032 0000FAB7 7410     <1>      jz       short sysvideo_27_4_0
11033                               <1>      ;mov      ax, 256
11034 0000FAB9 39C1     <1>      cmp      ecx, eax ; 256
11035 0000FABB 77C4     <1>      ja       short sysvideo_27_0
11036 0000FABD 48       <1>      dec      eax
11037 0000FABE 39C2     <1>      cmp      edx, eax ; 255
11038 0000FAC0 77BF     <1>      ja       short sysvideo_27_0
11039 0000FAC2 40       <1>      inc      eax
11040 0000FAC3 29D0     <1>      sub      eax, edx ; 256 - DX
11041 0000FAC5 39C8     <1>      cmp      eax, ecx
11042 0000FAC7 72B8     <1>      jb       short sysvideo_27_0
11043                               <1>
11044                               <1> sysvideo_27_4_0:
11045 0000FAC9 89F5     <1>      mov      ebp, esi
11046                               <1>
11047 0000FACB 80FB06   <1>      cmp      bl, 6
11048 0000FACE 7768     <1>      ja       short sysvideo_27_13
11049 0000FAD0 7210     <1>      jb       short sysvideo_27_5
11050                               <1>      ; bl = 6
11051 0000FAD2 F605[6E9D0100]10 <1>      test     byte [ufont], 16 ; 8x16 user font loaded ?
11052 0000FAD9 74A6     <1>      jz       short sysvideo_27_0
11053                               <1>      ; read 8x16 user defined font
11054 0000FADB BE00400900 <1>      mov      esi, VGAFONT16USER
11055 0000FAE0 EB0C     <1>      jmp      short sysvideo_27_6
11056                               <1> sysvideo_27_5:
11057 0000FAE2 80FB04   <1>      cmp      bl, 4
11058 0000FAE5 7239     <1>      jb       short sysvideo_27_11
11059 0000FAE7 7721     <1>      ja       short sysvideo_27_9
11060                               <1>      ; bl = 4
11061                               <1>      ; read 8x16 system font
11062 0000FAE9 BE[30630100] <1>      mov      esi, vgafont16
11063                               <1> sysvideo_27_6:
11064                               <1>      ; read 8x16 font
11065                               <1>      ;shl      dx, 4 ; * 16
11066                               <1>      ;shl      cx, 4 ; * 16 ; 16 bytes per char
11067                               <1>      ; 23/07/2022
11068 0000FAEE C1E204   <1>      shl      edx, 4 ; * 16
11069 0000FAF1 C1E104   <1>      shl      ecx, 4 ; * 16
11070                               <1> sysvideo_27_7:
11071 0000FAF4 89EF     <1>      mov      edi, ebp
11072                               <1>      ;add      edi, edx ; 16/02/2021
11073 0000FAF6 01D6     <1>      add      esi, edx
11074                               <1>      ; ecx = byte count
11075                               <1>      ; esi = source (in system memory)
11076                               <1>      ; edi = destination (in user memory)
11077 0000FAF8 E80E100000 <1>      call     transfer_to_user_buffer
11078 0000FAFD 7206     <1>      jc       short sysvideo_27_8
11079 0000FAFF 890D[548E0100] <1>      mov      [u.r0], ecx
11080                               <1> sysvideo_27_8:
11081 0000FB05 E974CFFFFF <1>      jmp      sysret
11082                               <1> sysvideo_27_9:
11083                               <1>      ; bl = 5
11084 0000FB0A F605[6E9D0100]08 <1>      test     byte [ufont], 8 ; 8x8 user font loaded ?
11085 0000FB11 74F2     <1>      jz       short sysvideo_27_8
11086                               <1>      ; read 8x8 user defined font
11087 0000FB13 BE00500900 <1>      mov      esi, VGAFONT8USER
11088                               <1> sysvideo_27_10:
11089                               <1>      ; read 8x8 font
11090                               <1>      ;shl      dx, 3 ; * 8
11091                               <1>      ;shl      cx, 3 ; * 8 ; 8 bytes per char
11092                               <1>      ; 23/07/2022
11093 0000FB18 C1E203   <1>      shl      edx, 3 ; * 8
11094 0000FB1B C1E103   <1>      shl      ecx, 3 ; * 8
11095 0000FB1E EBD4     <1>      jmp      short sysvideo_27_7
11096                               <1>
11097                               <1> sysvideo_27_11:
11098 0000FB20 80FB03   <1>      cmp      bl, 3 ; 8x14 system font
11099 0000FB23 720C     <1>      jb       short sysvideo_27_12 ; 8x8 system font
11100                               <1>      ; bl = 3
11101                               <1>      ; read 8x14 system font
11102                               <1>      ;mov      al, 14
11103                               <1>      ;mul      dl
11104                               <1>      ;mov      dx, ax
11105                               <1>      ;push     edx
11106                               <1>      ;mov      ax, 14
11107                               <1>      ;mul      cx
11108                               <1>      ;mov      cx, ax
11109                               <1>      ;pop      edx
11110 0000FB25 E8A5000000 <1>      call     sysvideo_27_14
11111 0000FB2A BE[30550100] <1>      mov      esi, vgafont14
11112 0000FB2F EBC3     <1>      jmp      short sysvideo_27_7
11113                               <1>
11114                               <1> sysvideo_27_12:
11115                               <1>      ; bl = 2
11116                               <1>      ; read 8x8 system font
11117 0000FB31 BE[304D0100] <1>      mov      esi, vgafont8
11118 0000FB36 EBE0     <1>      jmp      short sysvideo_27_10
11119                               <1>
11120                               <1> sysvideo_27_13:
11121                               <1>      ; overwrite font
11122 0000FB38 80FB0A   <1>      cmp      bl, 10
11123 0000FB3B 776E     <1>      ja       short sysvideo_27_22 ; 8x16 user font

```

```

11124 0000FB3D 7224      <1>      jb      short sysvideo_27_15
11125                    <1>      ; b1 = 10
11126 0000FB3F BF00500900 <1>      mov     edi, VGAFONT8USER
11127 0000FB44 F605[6E9D0100]08 <1>      test    byte [ufont], 8 ; 8x8 user font loaded ?
11128 0000FB4B 7556      <1>      jnz     short sysvideo_27_21 ; yes
11129 0000FB4D 08ED      <1>      or      ch, ch ; cx = 256
11130                    <1>      ;jnz     short sysvideo_27_21 ; 256 chars
11131 0000FB4F 7406      <1>      jz      short sysvideo_27_13_0
11132 0000FB51 66B90008 <1>      mov     cx, 8*256
11133 0000FB55 EB35      <1>      jmp     short sysvideo_27_18_0
11134                    <1> sysvideo_27_13_0:
11135                    <1>      ; copy system font to user font before overwrite
11136 0000FB57 BE[304D0100] <1>      mov     esi, vgafont8
11137                    <1>      ;push    edi
11138                    <1>      ;push    ecx
11139                    <1>      ;mov     cl, 64
11140                    <1>      ;rep     movsd
11141                    <1>      ;pop     ecx
11142                    <1>      ;pop     edi
11143                    <1>      ;mov     esi, ebp ; user's font buffer
11144 0000FB5C E880000000 <1>      call    sysvideo_27_23
11145 0000FB61 EB40      <1>      jmp     short sysvideo_27_21
11146                    <1>
11147                    <1> sysvideo_27_15:
11148                    <1>      ; check system font overwrite permission
11149 0000FB63 F605[6E9D0100]80 <1>      test    byte [ufont], 80h
11150 0000FB6A 7499      <1>      jz      short sysvideo_27_8
11151                    <1>
11152 0000FB6C 80FB08      <1>      cmp     b1, 8
11153 0000FB6F 773A      <1>      ja      short sysvideo_27_22 ; 8x16 system font
11154 0000FB71 722B      <1>      jb      short sysvideo_27_20 ; 8x8 system font
11155                    <1>      ; b1 = 8
11156                    <1>      ; overwrite 8x14 system font
11157                    <1>      ;mov     al, 14
11158                    <1>      ;mul     dl, 14
11159                    <1>      ;mov     dx, ax
11160                    <1>      ;push    edx
11161                    <1>      ;mov     ax, 14
11162                    <1>      ;mul     cx, ax
11163                    <1>      ;mov     cx, ax
11164                    <1>      ;pop     edx
11165 0000FB73 E857000000 <1>      call    sysvideo_27_14
11166 0000FB78 BF[30550100] <1>      mov     edi, vgafont14
11167 0000FB7D EB0B      <1>      jmp     short sysvideo_27_18
11168                    <1> sysvideo_27_16:
11169                    <1>      ; b1 = 9
11170                    <1>      ; overwrite 8x16 system font
11171 0000FB7F BF[30630100] <1>      mov     edi, vgafont16
11172                    <1> sysvideo_27_17:
11173                    <1>      ; overwrite 8x16 font
11174                    <1>      ;shl     dx, 4 ; * 16
11175                    <1>      ;shl     cx, 4 ; * 16 ; 16 bytes per char
11176                    <1>      ; 23/07/2022
11177 0000FB84 C1E204      <1>      shl     edx, 4 ; * 16
11178 0000FB87 C1E104      <1>      shl     ecx, 4 ; * 16
11179                    <1> sysvideo_27_18:
11180                    <1>      add     edi, edx
11181                    <1>      ;add     esi, edx ; 16/02/2021
11182                    <1> sysvideo_27_18_0:
11183                    <1>      ; ecx = byte count
11184                    <1>      ; esi = source (in user memory)
11185                    <1>      ; edi = destination (in system memory)
11186 0000FB8C E8C40F0000 <1>      call    transfer_from_user_buffer
11187 0000FB91 7206      <1>      jc      short sysvideo_27_19
11188 0000FB93 890D[548E0100] <1>      mov     [u.r0], ecx
11189                    <1> sysvideo_27_19:
11190 0000FB99 E9E0CEFFFF <1>      jmp     sysret
11191                    <1> sysvideo_27_20:
11192                    <1>      ; b1 = 7
11193                    <1>      ; overwrite 8x8 system font
11194 0000FB9E BF[304D0100] <1>      mov     edi, vgafont8
11195                    <1> sysvideo_27_21:
11196                    <1>      ; overwrite 8x8 font
11197                    <1>      ;shl     dx, 3 ; * 8
11198                    <1>      ;shl     cx, 3 ; * 8 ; 8 bytes per char
11199                    <1>      ; 23/07/2022
11200 0000FBA3 C1E203      <1>      shl     edx, 3 ; * 8
11201 0000FBA6 C1E103      <1>      shl     ecx, 3 ; * 8
11202 0000FBA9 EBD5      <1>      jmp     short sysvideo_27_18
11203                    <1> sysvideo_27_22:
11204                    <1>      ; b1 = 11
11205                    <1>      ; overwrite 8x16 user defined font
11206 0000FBAB BF00400900 <1>      mov     edi, VGAFONT16USER
11207 0000FBB0 F605[6E9D0100]10 <1>      test    byte [ufont], 16 ; 8x16 user font loaded ?
11208 0000FBB7 75CB      <1>      jnz     short sysvideo_27_17 ; yes
11209 0000FBB9 08ED      <1>      or      ch, ch ; cx = 256
11210                    <1>      ;jnz     short sysvideo_27_17 ; 256 chars
11211 0000FBBB 7406      <1>      jz      short sysvideo_27_22_0
11212 0000FBBD 66B90010 <1>      mov     cx, 16*256
11213 0000FBC1 EBC9      <1>      jmp     short sysvideo_27_18_0
11214                    <1> sysvideo_27_22_0:
11215                    <1>      ; copy system font to user font before overwrite
11216 0000FBC3 BE[30630100] <1>      mov     esi, vgafont16
11217                    <1>      ;push    edi
11218                    <1>      ;push    ecx
11219                    <1>      ;mov     cl, 64
11220                    <1>      ;rep     movsd
11221                    <1>      ;pop     ecx
11222                    <1>      ;pop     edi
11223                    <1>      ;mov     esi, ebp ; user's font buffer
11224 0000FBC8 E814000000 <1>      call    sysvideo_27_23
11225 0000FBCD EBB5      <1>      jmp     short sysvideo_27_17
11226                    <1>
11227                    <1> sysvideo_27_14:
11228                    <1>      ; 16/02/2021
11229 0000FBCF 52          <1>      push    edx
11230 0000FBD0 66B80E00 <1>      mov     ax, 14
11231 0000FBD4 66F7E1      <1>      mul     cx
11232 0000FBD7 89C1      <1>      mov     ecx, eax
11233 0000FBD9 5A          <1>      pop     edx
11234 0000FBDA B00E      <1>      mov     al, 14
11235 0000FBD8 F6E2      <1>      mul     dl
11236 0000FBDE 89C2      <1>      mov     edx, eax
11237 0000FBE0 C3          <1>      retn
11238                    <1>
11239                    <1>      ;mov     al, 14
11240                    <1>      ;mul     dl
11241                    <1>      ;mov     dx, ax
11242                    <1>      ;push    edx
11243                    <1>      ;; 12/02/2021
11244                    <1>      ;mov     ax, 14
11245                    <1>      ;;mov     eax, 14
11246                    <1>      ;;mul     cx
11247                    <1>      ;mul     ecx

```

```

11248 <1> ;mov cx, ax
11249 <1> ;mov ecx, eax
11250 <1> ;pop edx
11251 <1> ;retn
11252 <1>
11253 <1> sysvideo_27_23:
11254 0000FBE1 57 <1> push edi
11255 0000FBE2 51 <1> push ecx
11256 0000FBE3 B140 <1> mov cl, 64
11257 0000FBE5 F3A5 <1> rep movsd
11258 0000FBE7 59 <1> pop ecx
11259 0000FBE8 5F <1> pop edi
11260 0000FBE9 89EE <1> mov esi, ebp ; user's font buffer
11261 0000FBEB C3 <1> ret
11262 <1>
11263 <1> sysvideo_28:
11264 <1> ; 23/07/2022
11265 <1> ; 24/01/2021
11266 <1> ; 23/01/2021
11267 <1> ; 18/01/2021
11268 0000FBEC 80FF0E <1> cmp bh, 14
11269 <1> ;jb sysvideo_29
11270 <1> ; 23/07/2022
11271 0000FBEF 7222 <1> jnb short sysvideo_28_29
11272 <1> ;ja sysvideo_30
11273 <1> ; 23/07/2022
11274 0000FBF1 7725 <1> ja short sysvideo_28_30
11275 <1>
11276 <1> ; BH = 14
11277 <1> ; Save/Restore Super VGA video state
11278 <1>
11279 <1> ; BL = options
11280 <1> ; bit 0 - Save (0) or Restore (1)
11281 <1> ; bit 1 - controller hardware state
11282 <1> ; bit 2 - BIOS data state
11283 <1> ; bit 3 - DAC state
11284 <1> ; bit 4 - (extended) Register state
11285 <1> ; bit 5 - system (0) or user (1) memory
11286 <1> ; bit 6 - verify without transfer
11287 <1> ; bit 7 - not used (must be 0)
11288 <1>
11289 <1> ; ECX = Buffer address or VideoStateID
11290 <1>
11291 0000FBF3 803D[86090000]02 <1> cmp byte [vbe3], 2 ; VESA VBE2 or VBE3 ?
11292 0000FBFA 7721 <1> ja short sysvideo_28_0 ; yes
11293 0000FBFC 7210 <1> jb short sysvideo_28_16 ; not a SVGA sys !
11294 <1>
11295 <1> ; == VBE2 ==
11296 <1> ; Check Bochs/Qemu/VirtualBox PC emulator
11297 <1> ; (vbe2 is usable only for emulator's vbios)
11298 0000FBFE 8A25[87090000] <1> mov ah, [vbe2bios]
11299 0000FC04 80FCC0 <1> cmp ah, 0C0h
11300 0000FC07 7205 <1> jnb short sysvideo_28_16 ; unknown vbios !
11301 0000FC09 80FCC5 <1> cmp ah, 0C5h
11302 0000FC0C 760F <1> jna short sysvideo_28_0
11303 <1> ; Use kernel's vbios functions (video.s)
11304 <1> sysvideo_28_16:
11305 <1> ; unknown vbios !
11306 0000FC0E E96BCEFFFF <1> jmp sysret
11307 <1>
11308 <1> sysvideo_28_29:
11309 <1> ; 23/07/2022
11310 0000FC13 E955010000 <1> jmp sysvideo_29
11311 <1> sysvideo_28_30:
11312 <1> ; 23/07/2022
11313 0000FC18 E935020000 <1> jmp sysvideo_30
11314 <1>
11315 <1> sysvideo_28_0:
11316 0000FC1D 80FB7F <1> cmp bl, 7Fh
11317 0000FC20 77EC <1> ja short sysvideo_28_16 ; unknown options
11318 <1>
11319 0000FC22 88DA <1> mov dl, bl
11320 0000FC24 80E21F <1> and dl, 1Fh
11321 0000FC27 D0EA <1> shr dl, 1
11322 0000FC29 74E3 <1> jz short sysvideo_28_16 ; invalid !
11323 <1> ; DL = VBE Function 4F04h Save/Restore options
11324 <1> ; bit 0 : controller hardware state
11325 <1> ; bit 1 : BIOS data state
11326 <1> ; bit 2 : DAC state
11327 <1> ; bit 3 : (extended) Register state
11328 <1>
11329 0000FC2B F6C320 <1> test bl, 32 ; bit 5
11330 <1> ;jnz sysvideo_28_7 ; user buffer
11331 <1> ; 23/07/2022
11332 0000FC2E 7405 <1> jz short sysvideo_28_17
11333 0000FC30 E9B1000000 <1> jmp sysvideo_28_7
11334 <1>
11335 <1> sysvideo_28_17: ; 23/07/2022
11336 <1> ; source or destination is kernel/system buffer
11337 <1>
11338 0000FC35 803D[709D0100]00 <1> cmp byte [srvsf], 0 ; srs permission flag
11339 0000FC3C 76D0 <1> jna short sysvideo_28_16 ; not permitted
11340 <1>
11341 0000FC3E F6C301 <1> test bl, 1
11342 0000FC41 743A <1> jz short sysvideo_28_4 ; Save
11343 <1>
11344 <1> ; Restore
11345 0000FC43 3B0D[729D0100] <1> cmp ecx, [VideoStateID]
11346 0000FC49 75C3 <1> jne short sysvideo_28_16 ; not correct ID !
11347 <1>
11348 0000FC4B 0FB6CA <1> movzx ecx, dl
11349 0000FC4E 80CA80 <1> or dl, 80h
11350 0000FC51 3A15[719D0100] <1> cmp dl, [srvs0]
11351 0000FC57 75B5 <1> jne short sysvideo_28_16 ; not correct !
11352 <1>
11353 0000FC59 88DA <1> mov dl, bl
11354 <1>
11355 <1> ; ecx = cl = options
11356 0000FC5B E8173FFFFF <1> call vbe_srs_gbs
11357 <1> ; ebx = state buffer size (data size)
11358 <1>
11359 0000FC60 891D[548E0100] <1> mov [u.r0], ebx
11360 <1>
11361 0000FC66 F6C240 <1> test dl, 64 ; verify without transfer
11362 0000FC69 75A3 <1> jnz short sysvideo_28_16 ; yes
11363 <1>
11364 0000FC6B BE00580900 <1> mov esi, VBE3VIDEOSTATE
11365 0000FC70 BF00760900 <1> mov edi, VBE3SAVERESTOREBLOCK
11366 0000FC75 87D9 <1> xchg ecx, ebx
11367 0000FC77 F3A4 <1> rep movsb
11368 <1>
11369 0000FC79 88D9 <1> mov cl, bl
11370 <1>
11371 <1> ; 23/01/2021

```

```

11372 0000FC7B EB44      <1>      jmp      short sysvideo_28_10
11373                  <1>
11374                  <1> sysvideo_28_4:
11375 0000FC7D 53          <1>      push     ebx
11376                  <1>      ; 24/01/2021
11377 0000FC7E 31DB        <1>      xor      ebx, ebx ; 0 ; use kernel's buffer
11378 0000FC80 881D[719D0100] <1>      mov     [srvso], bl ; 0 ; invalidate
11379 0000FC86 891D[729D0100] <1>      mov     [VideoStateID], ebx ; 0 ; invalidate
11380 0000FC8C 0FB6CA      <1>      movzx   ecx, dl ; options
11381 0000FC8F 8201          <1>      mov     dl, 1 ; save state
11382 0000FC91 E890000000 <1>      call    sysvideo_28_11 ; 23/01/2021
11383                  <1>      ; Note: VBE3 BIOS data save option will be
11384                  <1>      ; disabled.. ; 24/01/2021
11385 0000FC96 89CA          <1>      mov     edx, ecx ; state (save) options
11386 0000FC98 5B          <1>      pop     ebx
11387                  <1>
11388 0000FC99 6683F84F      <1>      cmp     ax, 4Fh ; successful ?
11389 0000FC9D 7536          <1>      jne     short sysvideo_28_3 ; no !
11390                  <1>
11391 0000FC9F F6C340        <1>      test    bl, 64 ; verify without transfer
11392 0000FCA2 7536          <1>      jnz     short sysvideo_28_6 ; yes
11393                  <1>
11394                  <1>      ; ecx = cl = options
11395 0000FCA4 E8CE3EFFFF      <1>      call    vbe_srs_gbs
11396                  <1>      ; ebx = state buffer size (data size)
11397                  <1>
11398 0000FCA9 BE00760900      <1>      mov     esi, VBE3SAVERESTOREBLOCK
11399 0000FCAE BF00580900      <1>      mov     edi, VBE3VIDEOSTATE
11400 0000FCB3 89D9          <1>      mov     ecx, ebx
11401 0000FCB5 F3A4          <1>      rep     movsb
11402                  <1>
11403 0000FCB7 88D1          <1>      mov     cl, dl
11404 0000FCB9 80C980        <1>      or      cl, 80h ; SVGA (VESA VBE) flag
11405                  <1>      ;mov     [srvso], dl
11406                  <1>
11407 0000FCBC E908010000      <1>      jmp      sysvideo_28_15
11408                  <1>
11409                  <1>      ; 23/01/2021
11410                  <1> sysvideo_28_10:
11411                  <1>      ; CL = VESA VBE3 Save/Restore options
11412                  <1>
11413 0000FCC1 8202          <1>      mov     dl, 2 ; restore state
11414                  <1>
11415 0000FCC3 E85C000000      <1>      call    sysvideo_28_1
11416                  <1>
11417 0000FCC8 6683F84F      <1>      cmp     ax, 4Fh ; successful ?
11418 0000FCCC 7407          <1>      je      short sysvideo_28_3
11419                  <1>      ;jmp     short sysvideo_28_9
11420                  <1>
11421                  <1> sysvideo_28_9:
11422                  <1>      ; return zero size (error) to user
11423 0000FCCE 29C0          <1>      sub     eax, eax
11424                  <1> sysvideo_28_5:
11425 0000FCD0 A3[548E0100] <1>      mov     [u.r0], eax
11426                  <1> sysvideo_28_3:
11427 0000FCD5 E9A4CDFFFF      <1>      jmp     sysret
11428                  <1>
11429                  <1> sysvideo_28_6:
11430                  <1>      ; use timer ticks as VideoStateID
11431 0000FCDA A1[20770100] <1>      mov     eax, [TIMER_LH]
11432 0000FCDF 09C0          <1>      or      eax, eax
11433 0000FCE1 75ED          <1>      jnz     short sysvideo_28_5
11434 0000FCE3 40          <1>      inc     eax
11435 0000FCE4 EBEA          <1>      jmp     short sysvideo_28_5
11436                  <1>
11437                  <1> sysvideo_28_7:
11438                  <1>      ; save/restore to/from user buffer
11439                  <1>
11440                  <1>      ; 23/01/2021
11441 0000FCE6 89CE          <1>      mov     esi, ecx ; user's vstate buffer
11442 0000FCE8 BF00760900      <1>      mov     edi, VBE3SAVERESTOREBLOCK
11443                  <1>
11444 0000FCED 0FB6CA      <1>      movzx   ecx, dl ; VESA VBE func 4F04h options
11445                  <1>
11446                  <1>      ; source or destination is user buffer
11447 0000FCF0 F6C301        <1>      test    bl, 1
11448 0000FCF3 7444          <1>      jz      short sysvideo_28_12 ; Save
11449                  <1>
11450                  <1>      ; Restore
11451 0000FCF5 803D[709D0100]00 <1>      cmp     byte [srvsf], 0 ; srs permission flag
11452 0000FCFC 766A          <1>      jna     short sysvideo_28_14 ; not permitted
11453                  <1>
11454 0000FCFE 88DA          <1>      mov     dl, bl ; 'sysvideo' options
11455                  <1>
11456                  <1>      ; ecx = cl = options
11457 0000FD00 E8723EFFFF      <1>      call    vbe_srs_gbs
11458                  <1>      ; ebx = state buffer size (data size)
11459                  <1>
11460 0000FD05 891D[548E0100] <1>      mov     [u.r0], ebx ; transfer count
11461                  <1>
11462 0000FD0B F6C240        <1>      test    dl, 64 ; verify without transfer
11463 0000FD0E 7558          <1>      jnz     short sysvideo_28_14 ; yes
11464                  <1>
11465 0000FD10 6681FB0008      <1>      cmp     bx, 2048
11466 0000FD15 73B7          <1>      jnb     short sysvideo_28_9 ; invalid
11467                  <1>
11468 0000FD17 87D9          <1>      xchg     ecx, ebx
11469                  <1>      ; esi = user buffer
11470                  <1>      ; edi = VBE3SAVERESTOREBLOCK
11471                  <1>
11472 0000FD19 E8370E0000      <1>      call    transfer_from_user_buffer
11473 0000FD1E 72AE          <1>      jc      short sysvideo_28_9 ; error
11474                  <1>
11475 0000FD20 89D9          <1>      mov     ecx, ebx ; Function 4F04h options
11476 0000FD22 EB9D          <1>      jmp     short sysvideo_28_10 ; 23/01/2021
11477                  <1>
11478                  <1> sysvideo_28_1:
11479 0000FD24 31DB        <1>      xor      ebx, ebx ; 0 ; use kernel's buffer
11480                  <1> sysvideo_28_11:
11481                  <1>      ; 24/01/2021
11482 0000FD26 803D[86090000]03 <1>      cmp     byte [vbe3], 3
11483 0000FD2D 7405          <1>      je      short sysvideo_28_2
11484                  <1>
11485                  <1>      ; VESA VBE2 (BOCHS/QEMU/VBOX) video bios
11486 0000FD2F E9AC3DFFFF      <1>      jmp     _vbe_biosfn_save_restore_state
11487                  <1> sysvideo_28_2:
11488                  <1>      ;24/01/2021
11489                  <1>      ;mov     eax, 4F04h ; Save/Restore vstate
11490                  <1>      ; VESA VBE3 video bios
11491 0000FD34 E9221DFFFF      <1>      jmp     _vbe3_pmfn_save_restore_state
11492                  <1>
11493                  <1> sysvideo_28_12:
11494                  <1>      ; Save
11495                  <1>      ;mov     edi, VBE3SAVERESTOREBLOCK

```

```

11496 <1>
11497 <1> ;movzx ecx, dl ; options
11498 0000FD39 56 <1> push esi
11499 0000FD3A 53 <1> push ebx
11500 <1> ; 23/01/2021
11501 0000FD3B 8201 <1> mov dl, 1 ; save state
11502 0000FD3D E8E2FFFFFF <1> call sysvideo_28_1
11503 0000FD42 5A <1> pop edx ; 'sysvideo' options
11504 0000FD43 5F <1> pop edi ; user's video state buffer
11505 <1>
11506 0000FD44 6683F84F <1> cmp ax, 4Fh ; successful ?
11507 0000FD48 751E <1> jne short sysvideo_28_14 ; no !
11508 <1>
11509 <1> ; ecx = cl = options
11510 0000FD4A E8283EFFFF <1> call vbe_srs_gbs
11511 <1> ; ebx = state buffer size (data size)
11512 <1>
11513 0000FD4F 89D9 <1> mov ecx, ebx ; transfer count
11514 <1>
11515 0000FD51 F6C240 <1> test dl, 64 ; verify without transfer
11516 0000FD54 750C <1> jnz short sysvideo_28_13 ; yes
11517 <1>
11518 <1> ;mov edi, esi
11519 0000FD56 BE00760900 <1> mov esi, VBE3SAVERESTOREBLOCK
11520 0000FD5B E8AB0D0000 <1> call transfer_to_user_buffer
11521 0000FD60 7206 <1> jc short sysvideo_28_14
11522 <1> sysvideo_28_13:
11523 0000FD62 890D[548E0100] <1> mov [u.r0], ecx
11524 <1> sysvideo_28_14:
11525 0000FD68 E911CDFFFF <1> jmp sysret
11526 <1>
11527 <1> sysvideo_29:
11528 <1> ; 18/01/2021
11529 <1> ; BH = 13
11530 <1> ; Save/Restore std VGA video state
11531 <1>
11532 <1> ; b1 = 0..3
11533 <1> ; save to or restore from
11534 <1> ; system buffer, VBE3VIDEOSTATE
11535 <1> ; ECX = VideoStateID for restoring
11536 <1> ; b1 = 4..7
11537 <1> ; save to or restore from
11538 <1> ; user buffer pointed by ECX
11539 <1>
11540 0000FD6D 80FB03 <1> cmp b1, 3
11541 0000FD70 7776 <1> ja short sysvideo_29_6
11542 <1>
11543 <1> ; source or destination is kernel/system buffer
11544 <1>
11545 0000FD72 803D[709D0100]00 <1> cmp byte [srvsf], 0 ; srs permission flag
11546 0000FD79 7668 <1> jna short sysvideo_29_5 ; not permitted
11547 <1>
11548 0000FD7B F6C301 <1> test b1, 1
11549 0000FD7E 7437 <1> jz short sysvideo_29_2 ; Save
11550 <1>
11551 <1> ; Restore
11552 0000FD80 3B0D[729D0100] <1> cmp ecx, [VideoStateID]
11553 0000FD86 755B <1> jne short sysvideo_29_5 ; not correct ID !
11554 0000FD88 80FB01 <1> cmp b1, 1
11555 0000FD8B 7709 <1> ja short sysvideo_29_0
11556 <1> ; b1 = 1
11557 0000FD8D BB6E000000 <1> mov ebx, 110
11558 0000FD92 B103 <1> mov cl, 3 ; ctrl, vbiOS data
11559 0000FD94 EB07 <1> jmp short sysvideo_29_1
11560 <1> sysvideo_29_0:
11561 <1> ; b1 = 3
11562 0000FD96 BB72030000 <1> mov ebx, 882
11563 0000FD9B B107 <1> mov cl, 7 ; ctrl, vbiOS data, dac
11564 <1> sysvideo_29_1:
11565 0000FD9D 3A0D[719D0100] <1> cmp cl, [srvso]
11566 0000FDA3 753E <1> jne short sysvideo_29_5 ; not correct !
11567 <1>
11568 0000FDA5 BE00580900 <1> mov esi, VBE3VIDEOSTATE ; 22/01/2021
11569 0000FDAA E85C3FFFFFF <1> call biosfn_restore_video_state
11570 0000FDAF 891D[548E0100] <1> mov [u.r0], ebx ; video state size (bytes)
11571 <1> ; jmp sysret
11572 0000FDB5 EB2C <1> jmp short sysvideo_29_5
11573 <1> sysvideo_29_2:
11574 <1> ;mov esi, ecx
11575 0000FDB7 BF00580900 <1> mov edi, VBE3VIDEOSTATE
11576 <1>
11577 0000FDBC B107 <1> mov cl, 7 ; ctrl, vbiOS data, dac
11578 0000FDBE 08DB <1> or b1, b1
11579 0000FDC0 7502 <1> jnz short sysvideo_29_3 ; b1 = 2
11580 <1> ; b1 = 0
11581 0000FDC2 B103 <1> mov cl, 3 ; ctrl, vbiOS data
11582 <1> sysvideo_29_3:
11583 0000FDC4 E8DA3DFFFF <1> call biosfn_save_video_state
11584 <1> sysvideo_28_15:
11585 <1> ; use timer ticks as VideoStateID
11586 0000FDC9 A1[20770100] <1> mov eax, [TIMER_LH]
11587 0000FDC E 21C0 <1> and eax, eax
11588 0000FDD0 7501 <1> jnz short sysvideo_29_4
11589 0000FDD2 40 <1> inc eax
11590 <1> sysvideo_29_4:
11591 0000FDD3 880D[719D0100] <1> mov [srvso], cl
11592 0000FDD9 A3[729D0100] <1> mov [VideoStateID], eax
11593 0000FDD E A3[548E0100] <1> mov [u.r0], eax
11594 <1> sysvideo_29_5:
11595 0000FDE3 E996CCFFFF <1> jmp sysret
11596 <1>
11597 <1> sysvideo_29_6:
11598 0000FDE8 80FB07 <1> cmp b1, 7
11599 0000FDEB 77F6 <1> ja short sysvideo_29_5 ; invalid sub function
11600 <1>
11601 0000FDED 89CE <1> mov esi, ecx
11602 0000FDEF BF00760900 <1> mov edi, VBE3SAVERESTOREBLOCK
11603 <1>
11604 <1> ; source or destination is user buffer
11605 0000FDF4 F6C301 <1> test b1, 1
11606 0000FDF7 7434 <1> jz short sysvideo_29_9 ; Save
11607 <1>
11608 <1> ; Restore
11609 0000FDF9 803D[709D0100]00 <1> cmp byte [srvsf], 0 ; srs permission flag
11610 0000FE00 76E1 <1> jna short sysvideo_29_5 ; not permitted
11611 <1>
11612 <1> ;mov esi, ecx
11613 <1> ;mov edi, VBE3SAVERESTOREBLOCK
11614 <1>
11615 0000FE02 80FB07 <1> cmp b1, 7
11616 0000FE05 7409 <1> je short sysvideo_29_7
11617 <1> ; b1 = 5
11618 0000FE07 B303 <1> mov b1, 3
11619 0000FE09 B96E000000 <1> mov ecx, 110

```

```

11620 0000FE0E EB05      <1>      jmp      short sysvideo_29_8
11621                  <1> sysvideo_29_7:
11622                  <1>      ; b1 = 7
11623 0000FE10 B972030000 <1>      mov     ecx, 882
11624                  <1> sysvideo_29_8:
11625 0000FE15 E83B0D0000 <1>      call    transfer_from_user_buffer
11626 0000FE1A 72C7      <1>      jc      short sysvideo_29_5
11627 0000FE1C 890D[548E0100] <1>      mov     [u.r0], ecx
11628 0000FE22 88D9      <1>      mov     cl, b1 ; mov cl,7 (mov cl,3)
11629 0000FE24 89FE      <1>      mov     esi, edi ; VBE3SAVERESTOREBLOCK
11630                  <1>      ; cl = 3 or 7
11631 0000FE26 E8E03EFFFF <1>      call    biosfn_restore_video_state
11632 0000FE2B EBB6      <1>      jmp     sysvideo_29_5
11633                  <1>      ; jmp sysret
11634                  <1> sysvideo_29_9:
11635                  <1>      ; Save
11636                  <1>      ; mov edi, VBE3SAVERESTOREBLOCK
11637                  <1>
11638 0000FE2D 80FB06      <1>      cmp     b1, 6
11639 0000FE30 7409      <1>      je      short sysvideo_29_10
11640                  <1>      ; b1 = 4
11641 0000FE32 BB6E000000 <1>      mov     ebx, 110
11642 0000FE37 B103      <1>      mov     cl, 3 ; ctrl, vbiOS data
11643 0000FE39 EB07      <1>      jmp     short sysvideo_29_11
11644                  <1> sysvideo_29_10:
11645                  <1>      ; b1 = 6
11646 0000FE3B BB72030000 <1>      mov     ebx, 882
11647 0000FE40 B107      <1>      mov     cl, 7 ; ctrl, vbiOS data, dac
11648                  <1> sysvideo_29_11:
11649 0000FE42 E85C3DFFFF <1>      call    biosfn_save_video_state
11650                  <1>
11651 0000FE47 89D9      <1>      mov     ecx, ebx ; transfer count
11652 0000FE49 89F7      <1>      mov     edi, esi
11653 0000FE4B BE00760900 <1>      mov     esi, VBE3SAVERESTOREBLOCK
11654                  <1>
11655                  <1>      ; call transfer_to_user_buffer
11656                  <1>      ; jc short sysvideo_29_5
11657                  <1>      ; mov [u.r0], ecx ; transfer count
11658                  <1>      ; jmp sysret
11659                  <1>      ; jmp short sysvideo_29_5
11660                  <1>
11661 0000FE50 EB1A      <1>      jmp     short sysvideo_29_12
11662                  <1>
11663                  <1> sysvideo_30:
11664 0000FE52 80FF0F      <1>      cmp     bh, 15
11665 0000FE55 7722      <1>      ja      short sysvideo_31 ; invalid function
11666                  <1>
11667                  <1>      ; BH = 15
11668                  <1>      ; Copy VESA EDID to user's buffer
11669                  <1>
11670 0000FE57 803D[19420000]4F <1>      cmp     byte [edid], 4Fh
11671 0000FE5E 7519      <1>      jne     short sysvideo_31 ; not ready !
11672                  <1>
11673                  <1>      ; and ecx, ecx
11674                  <1>      ; jz short sysvideo_31
11675                  <1>
11676                  <1>      ; ecx = user's buffer address
11677 0000FE60 89CF      <1>      mov     edi, ecx
11678 0000FE62 BE[8C9C0100] <1>      mov     esi, edid_info
11679 0000FE67 B980000000 <1>      mov     ecx, 128 ; 128 bytes
11680                  <1> sysvideo_29_12:
11681 0000FE6C E89A0C0000 <1>      call    transfer_to_user_buffer
11682 0000FE71 7206      <1>      jc      short sysvideo_31
11683                  <1>
11684 0000FE73 890D[548E0100] <1>      mov     [u.r0], ecx ; EDID size, 128 bytes
11685                  <1> sysvideo_31:
11686 0000FE79 E900CCFFFF <1>      jmp     sysret
11687                  <1>
11688                  <1> sysexec:
11689                  <1>      ; 07/07/2025 - TRDOS 386 v2.0.10
11690                  <1>      ; 21/08/2024 - TRDOS 386 v2.0.9
11691                  <1>      ; 23/07/2022 - TRDOS 386 v2.0.5
11692                  <1>      ; 06/02/2022 - Retro UNIX 386 v1.2
11693                  <1>      ; 18/11/2017
11694                  <1>      ; 14/11/2017
11695                  <1>      ; 13/11/2017
11696                  <1>      ; 24/10/2016 - 04/01/2017
11697                  <1>      ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)
11698                  <1>      ; 23/06/2015 - 23/10/2015 (Retro UNIX 386 v1)
11699                  <1>      ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
11700                  <1>
11701                  <1>      ; 'sysexec' initiates execution of a file whose path name if
11702                  <1>      ; pointed to by 'name' in the sysexec call.
11703                  <1>      ; 'sysexec' performs the following operations:
11704                  <1>      ; 1. obtains i-number of file to be executed via 'namei'.
11705                  <1>      ; 2. obtains i-node of file to be executed via 'iget'.
11706                  <1>      ; 3. sets trap vectors to system routines.
11707                  <1>      ; 4. loads arguments to be passed to executing file into
11708                  <1>      ; highest locations of user's core
11709                  <1>      ; 5. puts pointers to arguments in locations immediately
11710                  <1>      ; following arguments.
11711                  <1>      ; 6. saves number of arguments in next location.
11712                  <1>      ; 7. initializes user's stack area so that all registers
11713                  <1>      ; will be zeroed and the PS is cleared and the PC set
11714                  <1>      ; to core when 'sysret' restores registers
11715                  <1>      ; and does an rti.
11716                  <1>      ; 8. initializes u.r0 and u.sp
11717                  <1>      ; 9. zeros user's core down to u.r0
11718                  <1>      ; 10. reads executable file from storage device into core
11719                  <1>      ; starting at location 'core'.
11720                  <1>      ; 11. sets u.break to point to end of user's code with
11721                  <1>      ; data area appended.
11722                  <1>      ; 12. calls 'sysret' which returns control at location
11723                  <1>      ; 'core' via 'rti' instruction.
11724                  <1>
11725                  <1>      ; Calling sequence:
11726                  <1>      ; sysexec; namep; argp
11727                  <1>      ; Arguments:
11728                  <1>      ; namep - points to pathname of file to be executed
11729                  <1>      ; argp - address of table of argument pointers
11730                  <1>      ; argp1... argpn - table of argument pointers
11731                  <1>      ; argp1:<...0> ... argpn:<...0> - argument strings
11732                  <1>      ; Inputs: (arguments)
11733                  <1>      ; Outputs: -
11734                  <1>      ; .....
11735                  <1>
11736                  <1>      ; Retro UNIX 386 v1 modification:
11737                  <1>      ; User application runs in it's own virtual space
11738                  <1>      ; which is isolated from kernel memory (and other
11739                  <1>      ; memory pages) via 80386 paging in ring 3
11740                  <1>      ; privilege mode. Virtual start address is always 0.
11741                  <1>      ; User's core memory starts at linear address 400000h
11742                  <1>      ; (the end of the 1st 4MB).
11743                  <1>

```

```

11744 <1> ; Retro UNIX 8086 v1 modification:
11745 <1> ; user/application segment and system/kernel segment
11746 <1> ; are different and sysenter/sysret/sysrele routines
11747 <1> ; are different (user's registers are saved to
11748 <1> ; and then restored from system's stack.)
11749 <1> ;
11750 <1> ; NOTE: Retro UNIX 8086 v1 'arg2' routine gets these
11751 <1> ; arguments which were in these registers;
11752 <1> ; but, it returns by putting the 1st argument
11753 <1> ; in 'u.namep' and the 2nd argument
11754 <1> ; on top of stack. (1st argument is offset of the
11755 <1> ; file/path name in the user's program segment.)
11756 <1> ;
11757 <1> ;call arg2
11758 <1> ; * name - 'u.namep' points to address of file/path name
11759 <1> ; in the user's program segment ('u.segmt')
11760 <1> ; with offset in BX register (as sysopen argument 1).
11761 <1> ; * argp - sysexec argument 2 is in CX register
11762 <1> ; which is on top of stack.
11763 <1> ;
11764 <1> ; jsr r0,arg2 / arg0 in u.namep,arg1 on top of stack
11765 <1> ;
11766 <1> ; 23/06/2015 (32 bit modifications)
11767 <1> ;
11768 <1> ;; 13/11/2017
11769 <1> ;;mov [u.namep], ebx ; argument 1
11770 <1> ; 18/10/2015
11771 0000FE7E 890D[548F0100] <1> mov [argv], ecx ; * ; argument 2
11772 <1> ;
11773 <1> ; 13/11/2017
11774 0000FE84 89DE <1> mov esi, ebx
11775 0000FE86 E83A1E0000 <1> call set_working_path_x
11776 0000FE8B 7319 <1> jnc short sysexec_0
11777 <1> ;
11778 <1> ;; 'bad command or file name'
11779 <1> ;mov eax, ERR_BAD_CMD_ARG ; 01h ; TRDOS 8086
11780 <1> ;
11781 <1> ; 'file not found !' error
11782 0000FE8D B802000000 <1> mov eax, ERR_NOT_FOUND ; 02h ; TRDOS 8086
11783 <1> sysexec_not_found_err:
11784 <1> sysexec_access_error:
11785 <1> sysexec_ext_error:
11786 0000FE92 A3[548E0100] <1> mov [u.r0], eax
11787 0000FE97 A3[C08E0100] <1> mov [u.error], eax
11788 0000FE9C E8F91E0000 <1> call reset_working_path
11789 0000FEA1 E9B8CBFFFF <1> jmp error
11790 <1>
11791 <1> sysexec_0:
11792 <1> ; 13/11/2017
11793 <1> ;mov esi, FindFile_Name
11794 0000FEA6 66B80018 <1> mov ax, 1800h ; Only files
11795 0000FEAA E8BB8AFFFF <1> call find_first_file
11796 0000FEAF 72E1 <1> jc short sysexec_not_found_err ; eax = 2
11797 <1> ;
11798 <1> ; check_file attributes
11799 <1> ; (attribute bits = 00ADVSHR) ; 18h = Directory+Volume
11800 <1> ; BL = Attributes byte
11801 <1>
11802 0000FEB1 F6C306 <1> test bl, 6 ; system file or hidden file (S+H)
11803 <1> ;jz short sysexec_0ext
11804 0000FEB4 7417 <1> jz short sysexec_1 ; yes
11805 <1> ;
11806 <1> ; 13/11/2017
11807 <1> ; /// TRDOS386 permission check for multiuser mode ///
11808 <1> ; SYSTEM file or HIDDEN file !!
11809 <1> ; (Only super user has permission to run this file.)
11810 <1> ;
11811 <1> ; ([u.uid]=0 for super user or root in multiuser mode)
11812 <1> ; ([u.uid]=0 for any users in singleuser mode)
11813 0000FEB6 803D[A68E0100]00 <1> cmp byte [u.uid], 0 ; Super User ([u.uid]=0) ?
11814 <1> ;jna short sysexec_0ext
11815 0000FEBD 760E <1> jna short sysexec_1 ; yes
11816 <1> ;
11817 <1> ; 'permission denied !' error
11818 0000FEBF B80B000000 <1> moveax, ERR_FILE_ACCESS ; 11 = ERR_PERM_DENIED
11819 0000FEC4 EBCC <1> jmp short sysexec_access_error
11820 <1>
11821 <1> sysexec_not_exf:
11822 <1> ; 'not executable file !' error
11823 0000FEC6 B816000000 <1> mov eax, ERR_NOT_EXECUTABLE
11824 0000FECB EBC5 <1> jmp sysexec_ext_error
11825 <1>
11826 <1> ;sysexec_0ext:
11827 <1> sysexec_1:
11828 <1> ; 18/11/2017
11829 0000FECF BE[44800100] <1> mov esi, FindFile_Name
11830 <1> ; 13/11/2017
11831 <1> ; check program file name extension
11832 <1> ; ('.PRG' for current TRDOS version)
11833 0000FED2 E8F9A3FFFF <1> call check_prg_filename_ext
11834 0000FED7 72ED <1> jc short sysexec_not_exf
11835 <1> ;
11836 <1> ; 18/11/2017
11837 0000FED9 3C50 <1> cmp al, 'P'
11838 0000FEDB 75E9 <1> jne short sysexec_not_exf
11839 <1> ;
11840 <1> ; '.PRG' extension is OK.
11841 <1> ; Only '.PRG' files are valid program files
11842 <1> ; for current TRDOS 386 version.
11843 <1> ;
11844 0000FEDD 8B15[70800100] <1> mov edx, [FindFile_DirEntry+DirEntry_FileSize]
11845 0000FEE3 66A1[68800100] <1> mov ax, [FindFile_DirEntry+DirEntry_FstClusHI]
11846 0000FEE9 C1E010 <1> shl eax, 16
11847 0000FEEC 66A1[6E800100] <1> mov ax, [FindFile_DirEntry+DirEntry_FstClusLO]
11848 <1> ; EAX = First Cluster number
11849 <1> ; EDX = File Size
11850 <1> ;
11851 0000FEF2 A3[5C8F0100] <1> mov [i], eax
11852 0000FEF7 8915[608F0100] <1> mov [i.size], edx
11853 <1> ;
11854 <1> ;sysexec_1:
11855 <1> ; 23/07/2022 - TRDOS 386 Kernel v2.0.5
11856 <1> ; 06/02/2022 - Retro UNIX 386 v1.2
11857 <1> ; 13/11/2017 - TRDOS 386 (TRDOS v2.0)
11858 <1> ; 24/06/2015 - 23/10/2015 (Retro UNIX 386 v1)
11859 <1> ; Moving arguments to the end of [u.upage]
11860 <1> ; (by regarding page borders in user's memory space)
11861 <1> ;
11862 <1> ; 10/10/2015
11863 <1> ; 21/07/2015
11864 <1> ;mov ebp, esp ; (**)
11865 <1> ; 18/10/2015
11866 <1> ;mov edi, ebp
11867 <1> ; 23/07/2022

```

```

11868 0000FEFD 89E7      <1>      mov     edi, esp ; (**)
11869 0000FEFF B900010000    <1>      mov     ecx, MAX_ARG_LEN ; 256
11870                                <1>      ;sub     edi, MAX_ARG_LEN ; 256
11871 0000FF04 29CF      <1>      sub     edi, ecx
11872 0000FF06 89FC      <1>      mov     esp, edi ; !**
11873 0000FF08 31C0      <1>      xor     eax, eax
11874 0000FF0A A3[808E0100]    <1>      mov     [u.nread], eax ; 0
11875                                <1>      ; [argc] must be cleared because previous 'sysexec'
11876                                <1>      ; may leave it with any value after an error))
11877                                <1>      ;mov     [argc], ax ; 0 ; 13/11/2017
11878                                <1>      ; 23/07/2022
11879 0000FF0F A3[508F0100]    <1>      mov     [argc], eax ; 0
11880 0000FF14 49          <1>      dec     ecx ; 256 - 1
11881 0000FF15 890D[7C8E0100] <1>      mov     [u.count], ecx ; MAX_ARG_LEN - 1 ; 255
11882                                <1>      ;mov     dword [u.count], MAX_ARG_LEN - 1 ; 255
11883                                <1>      sysexec_2:
11884 0000FF1B 8B35[548F0100] <1>      mov     esi, [argv] ; 18/10/2015
11885 0000FF21 E86E020000    <1>      call    get_argp
11886                                <1>      ;mov     ecx, 4
11887                                <1>      ; 23/07/2022
11888 0000FF26 31C9      <1>      xor     ecx, ecx
11889 0000FF28 B104      <1>      mov     cl, 4
11890                                <1>      sysexec_3:
11891 0000FF2A 21C0      <1>      and     eax, eax
11892 0000FF2C 7453      <1>      jz      short sysexec_6 ; 23/07/2022
11893                                <1>      ; 18/10/2015
11894 0000FF2E 010D[548F0100] <1>      add     [argv], ecx ; 4
11895                                <1>      ;;inc    word [argc]
11896                                <1>      ; 23/07/2022
11897                                <1>      ;inc     dword [argc]
11898 0000FF34 FE05[508F0100] <1>      inc     byte [argc]
11899                                <1>      ;
11900 0000FF3A A3[788E0100]    <1>      mov     [u.base], eax
11901                                <1>      ; 23/10/2015
11902 0000FF3F 66C705[B88E0100]00- <1>      mov     word [u.pcount], 0
11902 0000FF47 00          <1>
11903                                <1>      sysexec_4:
11904 0000FF48 E85F080000    <1>      call    cpass ; get a character from user's core memory
11905 0000FF4D 750B      <1>      jnz     short sysexec_5
11906                                <1>      ; (max. 255 chars + null)
11907                                <1>      ; 18/10/2015
11908 0000FF4F 28C0      <1>      sub     al, al
11909 0000FF51 AA          <1>      stosb
11910 0000FF52 FF05[808E0100] <1>      inc     dword [u.nread]
11911                                <1>      ; 23/07/2022
11912 0000FF58 EB27      <1>      jmp     short sysexec_6 ; 24/04/2016
11913                                <1>      sysexec_5:
11914 0000FF5A AA          <1>      stosb
11915 0000FF5B 20C0      <1>      and     al, al
11916 0000FF5D 75E9      <1>      jnz     short sysexec_4
11917                                <1>      ;mov     ecx, 4
11918                                <1>      ; 23/07/2022
11919 0000FF5F 31C9      <1>      xor     ecx, ecx
11920 0000FF61 B104      <1>      mov     cl, 4
11921 0000FF63 390D[4C8F0100] <1>      cmp     [ncount], ecx ; 4
11922 0000FF69 72B0      <1>      jb      short sysexec_2
11923 0000FF6B 8B35[488F0100] <1>      mov     esi, [nbase]
11924 0000FF71 010D[488F0100] <1>      add     [nbase], ecx ; 4
11925                                <1>      ;sub     [ncount], cx
11926                                <1>      ; 23/07/2022
11927 0000FF77 290D[4C8F0100] <1>      sub     [ncount], ecx
11928 0000FF7D 8B06      <1>      mov     eax, [esi]
11929 0000FF7F EBA9      <1>      jmp     short sysexec_3
11930                                <1>
11931                                <1>      sysexec_6:
11932                                <1>      ; 23/07/2022 - TRDOS 386 kernel v2.0.5
11933                                <1>      ; 19/11/2017
11934                                <1>      ; 18/11/2017
11935                                <1>      ; 14/11/2017
11936                                <1>      ; 13/11/2017
11937 0000FF81 8925[548F0100] <1>      mov     [argv], esp ; !** ; start address of argument list
11938                                <1>
11939                                <1>      ; 04/01/2017
11940                                <1>      ; 24/10/2016
11941                                <1>      ; 02/05/2016
11942                                <1>      ; 23/04/2016 (TRDOS 386)
11943                                <1>      ; 18/10/2015 ('sysexec_6')
11944                                <1>      ; 23/06/2015
11945 0000FF87 A1[AC8E0100]    <1>      mov     eax, [u.pgdir] ; physical address of page directory
11946                                <1>      ;cmp     eax, [k_page_dir] ; TRDOS MainProg ?
11947                                <1>      ;je      short sysexec_7
11948                                <1>      ; 19/11/2017
11949 0000FF8C 8B1D[B08E0100] <1>      mov     ebx, [u.ppgdir] ; phy addr of the parent's page dir
11950 0000FF92 E8BF59FFFF    <1>      call    deallocate_page_dir
11951                                <1>      sysexec_7:
11952 0000FF97 E8EF58FFFF    <1>      call    make_page_dir
11953                                <1>      ;jc      panic ; allocation error
11954                                <1>      ; ; after a deallocation would be nonsense !?
11955                                <1>      ; 23/07/2022
11956 0000FF9C 7243      <1>      jc      short sysexec_panic
11957                                <1>
11958                                <1>      ; 24/07/2015
11959                                <1>      ; map kernel pages (1st 4MB) to PDE 0
11960                                <1>      ; of the user's page directory
11961                                <1>      ; (It is needed for interrupts!)
11962                                <1>      ; 18/10/2015
11963 0000FF9E 8B15[A0760100] <1>      mov     edx, [k_page_dir] ; kernel's page directory
11964 0000FFA4 8B02      <1>      mov     eax, [edx] ; physical address of
11965                                <1>      ; kernel's first page table (1st 4 MB)
11966                                <1>      ; (PDE 0 of kernel's page directory)
11967 0000FFA6 8B15[AC8E0100] <1>      mov     edx, [u.pgdir]
11968 0000FFAC 8902      <1>      mov     [edx], eax ; PDE 0 (1st 4MB)
11969                                <1>      ;
11970                                <1>      ; 20/07/2015
11971 0000FFAE BB00004000    <1>      mov     ebx, CORE ; start address = 0 (virtual) + CORE
11972                                <1>      ; 18/10/2015
11973 0000FFB3 BE[408F0100] <1>      mov     esi, pcORE ; physical start address
11974                                <1>      sysexec_8:
11975 0000FFB8 B907000000    <1>      mov     ecx, PDE_A_USER + PDE_A_WRITE + PDE_A_PRESENT
11976 0000FFBD E8E758FFFF    <1>      call    make_page_table
11977                                <1>      ;jc      panic
11978                                <1>      ; 23/07/2022
11979 0000FFC2 721D      <1>      jc      short sysexec_panic
11980                                <1>      ;
11981                                <1>      ;mov     ecx, PTE_A_USER + PTE_A_WRITE + PTE_A_PRESENT
11982 0000FFC4 E8EE58FFFF    <1>      call    make_page ; make new page, clear and set the pte
11983                                <1>      ;jc      panic
11984                                <1>      ; 23/07/2022
11985 0000FFC9 7216      <1>      jc      short sysexec_panic
11986                                <1>      ;
11987 0000FFCB 8906      <1>      mov     [esi], eax ; 24/06/2015
11988                                <1>      ; ebx = virtual address (24/07/2015)
11989                                <1>      ; 23/07/2022
11990                                <1>      ;call    add_to_swap_queue

```



```

11991      <1>      ; 18/10/2015
11992 0000FFCD 81FE[448F0100] <1>      cmp     esi, ecore ; user's stack (last) page ?
11993 0000FFD3 7411 <1>      je      short sysexec_9 ; yes
11994 0000FFD5 BE[448F0100] <1>      mov     esi, ecore ; physical address of the last page
11995      <1>      ; 20/07/2015
11996 0000FFDA BB00F0FFFF <1>      mov     ebx, (ECORE - PAGE_SIZE) + CORE
11997      <1>      ; ebx = virtual end address + segment base address - 4K
11998 0000FFDF EBD7 <1>      jmp     short sysexec_8
11999      <1>      sysexec_panic:
12000      <1>      ; 23/07/2022
12001 0000FFE1 E9236EFFFF <1>      jmp     panic
12002      <1>
12003      <1>      sysexec_9:
12004      <1>      ; 23/07/2022 - TRDOS 386 kernel v2.0.5
12005      <1>      ; 19/11/2017
12006      <1>      ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)
12007      <1>      ; 25/06/2015 - 26/08/2015 - 18/10/2015
12008      <1>      ; move arguments from kernel stack to [ecore]
12009      <1>      ; (argument list/line will be copied from kernel stack
12010      <1>      ; frame to the last (stack) page of user's core memory)
12011      <1>      ; 18/10/2015
12012 0000FFE6 8B3D[448F0100] <1>      mov     edi, [ecore]
12013 0000FFEC 81C700100000 <1>      add     edi, PAGE_SIZE
12014      <1>      ; 19/11/2017
12015      <1>      ; sub     edi, 4
12016      <1>      ; mov     dword [edi], 0
12017      <1>      ; mov     ebx, edi
12018      <1>      ;
12019      <1>      ; movzx   eax, word [argc]
12020      <1>      ; or      eax, eax
12021      <1>      ; jz      short sysexec_13 ; 19/11/2017
12022      <1>      ; jnz     short sysexec_10
12023      <1>      ; mov     ebx, edi
12024      <1>      ; sub     ebx, 4
12025      <1>      ; mov     [ebx], eax ; 0
12026      <1>      ; jmp     short sysexec_13
12027      <1>      ; 23/07/2022
12028      <1>      ; [argc] < 32
12029 0000FFF2 A1[508F0100] <1>      mov     eax, [argc]
12030 0000FFF7 09C0 <1>      or      eax, eax
12031 0000FFF9 7509 <1>      jnz     short sysexec_10
12032 0000FFFB 89FB <1>      mov     ebx, edi
12033 0000FFFD 83EB04 <1>      sub     ebx, 4
12034 00010000 8903 <1>      mov     [ebx], eax ; 0
12035 00010002 EB47 <1>      jmp     short sysexec_13
12036      <1>
12037      <1>      sysexec_10:
12038 00010004 8B0D[808E0100] <1>      mov     ecx, [u.nread]
12039      <1>      ; 13/11/2017
12040      <1>      ; mov     esi, TextBuffer ; 'load_and_execute_file'
12041      <1>      ; mov     esi, esp ; 'sysexec'
12042 0001000A 8B35[548F0100] <1>      mov     esi, [argv] ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)
12043      <1>      ; 23/07/2022
12044 00010010 29CF <1>      sub     edi, ecx ; page end address - argument list length
12045      <1>      ; sub     ebx, ecx ; 19/11/2017
12046      <1>
12047      <1>      ;;;;
12048      <1>      ; 23/07/2022
12049      <1>      ; (move edi -backward- to dword boundary)
12050      <1>      ; ((this will prevent 'general protection fault' error
12051      <1>      ; as result of a lodsd or dword move instruction
12052      <1>      ; at the end of argument list))
12053 00010012 83EF03 <1>      sub     edi, 3
12054 00010015 83E7FC <1>      and     edi, ~3 ; (*)
12055      <1>      ;;;
12056      <1>
12057 00010018 89C2 <1>      mov     edx, eax
12058 0001001A FEC2 <1>      inc     dl ; argument count + 1 for argc value
12059 0001001C C0E202 <1>      shl     dl, 2 ; 4 * (argument count + 1)
12060      <1>      ; edx <= 128
12061 0001001F 89FB <1>      mov     ebx, edi
12062      <1>      ; mov     edi, ebx ; 19/11/2017
12063      <1>      ; 23/07/2022 (*) - edi is already dword aligned -
12064      <1>      ; and     bl, 0FCh ; 32 bit (dword) alignment
12065 00010021 29D3 <1>      sub     ebx, edx
12066 00010023 89FA <1>      mov     edx, edi
12067 00010025 F3A4 <1>      rep     movsb
12068 00010027 89D6 <1>      mov     esi, edx
12069 00010029 89DF <1>      mov     edi, ebx
12070 0001002B BA00F0BFFF <1>      mov     edx, ECORE - PAGE_SIZE ; virtual addr. of the last page
12071 00010030 2B15[448F0100] <1>      sub     edx, [ecore] ; difference (virtual - physical)
12072 00010036 AB <1>      stosd   ; eax = argument count
12073      <1>      sysexec_11:
12074 00010037 89F0 <1>      mov     eax, esi
12075 00010039 01D0 <1>      add     eax, edx
12076 0001003B AB <1>      stosd   ; eax = virtual address
12077      <1>      ; 23/07/2022
12078 0001003C FE0D[508F0100] <1>      dec     byte [argc]
12079      <1>      ; dec     dword [argc]
12080      <1>      ; dec     word [argc] ; 14/11/2017
12081 00010042 7407 <1>      jz      short sysexec_13
12082      <1>      sysexec_12:
12083 00010044 AC <1>      lodsb
12084 00010045 20C0 <1>      and     al, al
12085 00010047 75FB <1>      jnz     short sysexec_12
12086 00010049 EBEC <1>      jmp     short sysexec_11
12087      <1>      sysexec_13:
12088      <1>      ; 24/10/2016
12089      <1>      ; 24/04/2016 - TRDOS 386 (TRDOS v2.0)
12090      <1>      ; 23/06/2015 - 19/10/2015 (Retro UNIX 386 v1, 'sysexec_13')
12091      <1>      ;
12092      <1>      ; moving arguments to [ecore] is OK here..
12093      <1>      ;
12094      <1>      ; ebx = beginning address of argument list pointers
12095      <1>      ; in user's stack
12096 0001004B 2B1D[448F0100] <1>      sub     ebx, [ecore]
12097 00010051 81C300F0BFFF <1>      add     ebx, (ECORE - PAGE_SIZE)
12098      <1>      ; end of core - 4096 (last page)
12099      <1>      ; (virtual address)
12100 00010057 891D[548F0100] <1>      mov     [argv], ebx
12101 0001005D 891D[848E0100] <1>      mov     [u.break], ebx ; available user memory
12102      <1>      ;
12103 00010063 29C0 <1>      sub     eax, eax
12104 00010065 C705[7C8E0100]2000- <1>      mov     dword [u.count], 32 ; Executable file header size
12104 0001006D 0000 <1>
12105 0001006F C705[688E0100]- <1>      mov     dword [u.fofp], u.off
12105 00010075 [748E0100] <1>
12106 00010079 A3[748E0100] <1>      mov     [u.off], eax ; 0
12107 0001007E A3[788E0100] <1>      mov     [u.base], eax ; 0, start of user's core (virtual)
12108      <1>      ; 24/10/2016
12109 00010083 A0[62770100] <1>      mov     al, [Current_Drv]
12110 00010088 A2[398E0100] <1>      mov     [cdev], al
12111      <1>      ;
12112 0001008D A1[5C8F0100] <1>      mov     eax, [ii] ; Fist cluster of the Program (PRG) file

```

```

12113      <1>      ; EAX = First cluster of the executable file
12114 00010092 E8DF050000 <1>      call      readi
12115      <1>
12116 00010097 8B0D[848E0100] <1>      mov     ecx, [u.break] ; top of user's stack (physical addr.)
12117 0001009D 890D[7C8E0100] <1>      mov     [u.count], ecx ; save for overrun check
12118      <1>
12119 000100A3 8B0D[808E0100] <1>      mov     ecx, [u.nread]
12120 000100A9 890D[848E0100] <1>      mov     [u.break], ecx ; virtual address (offset from start)
12121 000100AF 80F920 <1>      cmp     cl, 32
12122 000100B2 7540 <1>      jne     short sysexec_15
12123      <1>
12124      <1>      ; Retro UNIX 386 v1 (32 bit) executable file header format
12125 000100B4 8B35[408F0100] <1>      mov     esi, [pcore] ; start address of user's core memory
12126      <1>      ; (phys. start addr. of the exec. file)
12127 000100BA AD <1>      lodsd
12128 000100BB 663DEB1E <1>      cmp     ax, 1EEBh ; EBH, 1Eh -> jump to +32
12129 000100BF 7533 <1>      jne     short sysexec_15
12130 000100C1 AD <1>      lodsd
12131 000100C2 89C1 <1>      mov     ecx, eax ; text (code) section size
12132 000100C4 AD <1>      lodsd
12133 000100C5 01C1 <1>      add     ecx, eax ; + data section size (initialized data)
12134 000100C7 89CB <1>      mov     ebx, ecx
12135 000100C9 AD <1>      lodsd
12136 000100CA 01C3 <1>      add     ebx, eax ; + bss section size (for overrun checking)
12137 000100CC 3B1D[7C8E0100] <1>      cmp     ebx, [u.count]
12138 000100D2 7711 <1>      ja     short sysexec_14 ; program overruns stack !
12139      <1>
12140      <1>      ; add bss section size to [u.break]
12141 000100D4 0105[848E0100] <1>      add     [u.break], eax
12142      <1>
12143 000100DA 83E920 <1>      sub     ecx, 32 ; header size (already loaded)
12144      <1>      ; cmp     ecx, [u.count]
12145      <1>      ; jnb     short sysexec_16
12146 000100DD 890D[7C8E0100] <1>      mov     [u.count], ecx ; required read count
12147 000100E3 EB29 <1>      jmp     short sysexec_16
12148      <1>
12149      <1>      sysexec_14:
12150 000100E5 C705[C08E0100]0400- <1>      ; insufficient (out of) memory
12150 000100ED 0000 <1>      mov     dword [u.error], ERR_MINOR_IM ; 1
12151 000100EF E96AC9FFFF <1>      jmp     error
12152      <1>
12153 000100F4 8B15[608F0100] <1>      sysexec_15:
12154 000100FA 29CA <1>      mov     edx, [i.size] ; file size
12155 000100FC 7626 <1>      sub     edx, ecx ; file size - loaded bytes
12156 000100FE 01D1 <1>      jna     short sysexec_17 ; no need to next read
12157 00010100 3B0D[7C8E0100] <1>      add     ecx, edx ; [i.size]
12158 00010106 77DD <1>      cmp     ecx, [u.count] ; overrun check (!)
12159 00010108 8915[7C8E0100] <1>      ja     short sysexec_14
12160      <1>      mov     [u.count], edx
12161      <1>      sysexec_16:
12162 0001010E A1[5C8F0100] <1>      mov     eax, [ii] ; first cluster
12163 00010113 E85E050000 <1>      call     readi
12164 00010118 8B0D[808E0100] <1>      mov     ecx, [u.nread]
12165 0001011E 010D[848E0100] <1>      add     [u.break], ecx
12166      <1>      sysexec_17:
12167      <1>      ; 07/07/2025
12168      <1>      ; mov     eax, [ii] ; first cluster
12169 00010124 31C0 <1>      ; call    iclose
12170      <1>      xor     eax, eax
12171      <1>
12172      <1>      ; 21/08/2024
12173      <1>      ; inc     al
12174      <1>      ; mov     [u.intr], ax ; 1 (interrupt/time-out is enabled)
12175      <1>      ; mov     [u.quit], ax ; 1 ('ctrl+brk' signal is enabled)
12176      <1>      ; 23/07/2022
12177      <1>      ; dec     al
12178      <1>      ; 21/08/2024
12179 00010126 48 <1>      dec     eax
12180 00010127 66A3[A08E0100] <1>      mov     [u.intr], ax ; -1 ; 0FFFFh ; enable CTRL+CRK
12181 0001012D 40 <1>      inc     eax
12182 0001012E 66A3[A28E0100] <1>      mov     [u.quit], ax ; 0 ; reset CTRL+BRK flag
12183      <1>
12184 00010134 3905[B08E0100] <1>      ; cmp     dword [u.ppgdir], 0 ; is the caller MainProg (kernel) ?
12185 0001013A 770C <1>      cmp     [u.ppgdir], eax ; 0 ; 23/07/2022
12186      <1>      ja     short sysexec_18 ; no, the caller is user process
12187 0001013C 8B15[A0760100] <1>      ; If the caller is kernel (MainProg), 'sysexec' will come here
12188 00010142 8915[B08E0100] <1>      mov     edx, [k_page_dir] ; kernel's page directory
12189      <1>      mov     [u.ppgdir], edx ; next time 'sysexec' must not come here
12190      <1>      sysexec_18:
12191      <1>      ; 02/05/2016
12192      <1>      ; 24/04/2016 (TRDOS 386 = TRDOS v2.0)
12193      <1>      ; 18/10/2015 (Retro UNIX 386 v1)
12194      <1>      ; 05/08/2015
12195      <1>      ; 29/07/2015
12196      <1>      ;
12197      <1>      ; **** arguments list test start - 19/11/2017
12198      <1>      ; mov     ebp, [argv]
12199      <1>      ; sub     ebp, ECORE - 4096
12200      <1>      ; add     ebp, [ecore]
12201      <1>      ;
12202      <1>      ; mov     ebx, [ebp]
12203      <1>      ; mov     [argc], bx
12204      <1>      ; add     ebp, 4
12205      <1>      ; mov     byte [ccolor], 1Fh
12206      <1>      ;_zx0:
12207      <1>      ; cmp     word [argc], 0
12208      <1>      ; jna     short _zx2
12209      <1>      ;_zx1:
12210      <1>      ; push    ebp
12211      <1>      ; mov     esi, [ebp]
12212      <1>      ;
12213      <1>      ; sub     esi, ECORE - 4096
12214      <1>      ; add     esi, [ecore]
12215      <1>      ; call    print_cmsg
12216      <1>      ;
12217      <1>      ; dec     word [argc]
12218      <1>      ; jz     short _zx2
12219      <1>      ;
12220      <1>      ; mov     al, '.'
12221      <1>      ; mov     bl, 07h
12222      <1>      ; mov     bh, [u.ttyn]
12223      <1>      ; call    _write_tty
12224      <1>      ;
12225      <1>      ; pop     ebp
12226      <1>      ; add     ebp, 4
12227      <1>      ; jmp     short _zx1
12228      <1>      ;_zx2:
12229      <1>      ; pop     ebp
12230      <1>      ; mov     byte [ccolor], 07h
12231      <1>      ; mov     eax, 1
12232      <1>      ;
12233      <1>      ; **** arguments list test stop
12234      <1>      ; Test result is OK! (there is not a wrong thing) - 19/11/2017
12235 00010148 8B2D[548F0100] <1>      mov     ebp, [argv] ; user's stack pointer must point to argument

```

```

12236                                     <1>             ; list pointers (argument count)
12237 0001014E FA                         <1> cli
12238 0001014F 8B25[3C760100]            <1> mov     esp, [tss.esp0] ; ring 0 (kernel) stack pointer
12239                                     <1> ;mov     esp, [u.sp] ; Restore kernel stack
12240                                     <1> ; for this process
12241                                     <1> ;add     esp, 20 ; --> EIP, CS, EFLAGS, ESP, SS
12242                                     <1> ;xor     eax, eax ; 0
12243                                     <1> ; 23/07/2022
12244                                     <1> ;dec     al ; eax = 0
12245                                     <1> ; eax = 0
12246                                     <1>
12247                                     <1> ;mov     edx, UDATA
12248                                     <1> ; 18/11/2017
12249 00010155 6A23                       <1> push    UDATA ; user's stack segment
12250                                     <1> ;push    edx
12251 00010157 55                         <1> push    ebp ; user's stack pointer
12252                                     <1> ; (points to number of arguments)
12253                                     <1>
12254                                     <1> ; 04/01/2017
12255                                     <1> ; MainProg comes here while [sysflg]= 0FFh
12256                                     <1> ; (but sysexec comes here while [sysflg]= 0)
12257 00010158 C605[488E0100]00          <1> mov     byte [sysflg], 0 ; 04/01/2017
12258                                     <1> ; (timer_int sysflg control)
12259 0001015F FB                         <1> sti
12260 00010160 9C                         <1> pushfd ; EFLAGS
12261                                     <1> ; Set IF for enabling interrupts in user mode
12262                                     <1> ;or      dword [esp], 200h
12263                                     <1> ;
12264                                     <1> ;mov     bx, UCODE
12265                                     <1> ;push    bx ; user's code segment
12266 00010161 6A1B                       <1> push    UCODE
12267                                     <1> ;push    0
12268 00010163 50                         <1> push    eax ; EIP (=0) - start address -
12269 00010164 8925[4C8E0100]            <1> mov     [u.sp], esp ; 29/07/2015
12270                                     <1> ; 05/08/2015
12271                                     <1> ; Remedy of a General Protection Fault during 'iretd' is here !
12272                                     <1> ; ('push dx' would cause to general protection fault,
12273                                     <1> ; after 'pop ds' etc.)
12274                                     <1> ;
12275                                     <1> ; push dx ; ds (UDATA)
12276                                     <1> ; push dx ; es (UDATA)
12277                                     <1> ; push dx ; fs (UDATA)
12278                                     <1> ; push dx ; gs (UDATA)
12279                                     <1> ;
12280                                     <1> ; This is a trick to prevent general protection fault
12281                                     <1> ; during 'iretd' intrusion at the end of 'sysrele' (in u1.s):
12282 0001016A 66BA2300                   <1> mov     dx, UDATA ; 19/11/2017
12283 0001016E 8EC2                       <1> mov     es, dx ; UDATA
12284 00010170 06                         <1> push    es ; ds (UDATA)
12285 00010171 06                         <1> push    es ; es (UDATA)
12286 00010172 06                         <1> push    es ; fs (UDATA)
12287 00010173 06                         <1> push    es ; gs (UDATA)
12288 00010174 66BA1000                   <1> mov     dx, KDATA
12289 00010178 8EC2                       <1> mov     es, dx
12290                                     <1> ;
12291                                     <1> ; pushad simulation
12292 0001017A 89E5                       <1> mov     ebp, esp ; esp before pushad
12293 0001017C 50                         <1> push    eax ; eax (0)
12294 0001017D 50                         <1> push    eax ; ecx (0)
12295 0001017E 50                         <1> push    eax ; edx (0)
12296 0001017F 50                         <1> push    eax ; ebx (0)
12297 00010180 55                       <1> push    ebp ; esp before pushad
12298 00010181 50                         <1> push    eax ; ebp (0)
12299 00010182 50                         <1> push    eax ; esi (0)
12300 00010183 50                         <1> push    eax ; edi (0)
12301                                     <1> ;
12302 00010184 A3[548E0100]                <1> mov     [u.r0], eax ; eax = 0
12303 00010189 8925[508E0100]            <1> mov     [u.usp], esp
12304                                     <1>
12305                                     <1> ; 14/11/2017
12306 0001018F E9ECC8FFFF               <1> jmp     sysret0
12307                                     <1>
12308                                     <1> get_argp:
12309                                     <1> ; 08/08/2022
12310                                     <1> ; 23/07/2022 - TRDOS 386 kernel v2.0.5
12311                                     <1> ; 11/12/2021 - Retro UNIX 386 v1.2
12312                                     <1> ; 14/11/2017 - TRDOS 386 (TRDOS v2.0)
12313                                     <1> ; 18/10/2015 (nbase, ncount)
12314                                     <1> ; 21/07/2015
12315                                     <1> ; 24/06/2015 (Retro UNIX 386 v1)
12316                                     <1> ; Get (virtual) address of argument from user's core memory
12317                                     <1> ;
12318                                     <1> ; INPUT:
12319                                     <1> ;     esi = virtual address of argument pointer
12320                                     <1> ; OUTPUT:
12321                                     <1> ;     eax = virtual address of argument
12322                                     <1> ;
12323                                     <1> ; Modified registers: EAX, EBX, ECX, EDX, ESI
12324                                     <1> ;
12325 00010194 833D[B08E0100]00          <1> cmp     dword [u.ppgdir], 0 ; /etc/init ?
12326                                     <1> ; (the caller is kernel)
12327                                     <1> ;jna     short get_argpk
12328                                     <1> ; 08/08/2022
12329 0001019B 7705                       <1> ja      short get_argp5
12330 0001019D E985000000                 <1> jmp     get_argpk
12331                                     <1> get_argp5:
12332 000101A2 89F3                       <1> mov     ebx, esi
12333 000101A4 E85B5AFFFF                 <1> call    get_physical_addr ; get physical address
12334 000101A9 7253                       <1> jc      short get_argp_err ; 23/07/2022
12335 000101AB A3[488F0100]                <1> mov     [nbase], eax ; physical address
12336                                     <1> ;mov     [ncount], cx ; remain byte count in page (1-4096)
12337                                     <1> ; 23/07/2022
12338 000101B0 890D[4C8F0100]            <1> mov     [ncount], ecx
12339                                     <1> ;mov     eax, 4 ; 21/07/2015
12340 000101B6 31C0                       <1> xor     eax, eax
12341 000101B8 B004                       <1> mov     al, 4
12342                                     <1> ;cmp     cx, ax ; 4
12343                                     <1> ; 23/07/2022
12344 000101BA 39C1                       <1> cmp     ecx, eax ; 4
12345 000101BC 7354                       <1> jnb     short get_argp2
12346 000101BE 89F3                       <1> mov     ebx, esi
12347 000101C0 01CB                       <1> add     ebx, ecx
12348 000101C2 E83D5AFFFF                 <1> call    get_physical_addr ; get physical address
12349 000101C7 7235                       <1> jc      short get_argp_err
12350                                     <1> ;push    esi
12351 000101C9 89C6                       <1> mov     esi, eax
12352                                     <1> ;xchg    cx, [ncount]
12353                                     <1> ; 23/07/2022
12354 000101CB 870D[4C8F0100]            <1> xchg    ecx, [ncount]
12355 000101D1 8735[488F0100]            <1> xchg    esi, [nbase]
12356 000101D7 B504                       <1> mov     ch, 4
12357 000101D9 28CD                       <1> sub     ch, cl
12358                                     <1> get_argp0:
12359 000101DB AC                         <1> lodsb

```

```

12360 <1> ;push ax
12361 <1> ; 23/07/2022
12362 000101DC 50 <1> push eax
12363 000101DD FEC9 <1> dec cl
12364 000101DF 75FA <1> jnz short get_argp0
12365 000101E1 8B35[488F0100] <1> mov esi, [nbase]
12366 <1> ; 21/07/2015
12367 000101E7 0FB6C5 <1> movzx eax, ch
12368 000101EA 0105[488F0100] <1> add [nbase], eax
12369 <1> ;sub [ncount], ax
12370 <1> ; 23/07/2022
12371 000101F0 2905[4C8F0100] <1> sub [ncount], eax
12372 <1> get_argp1:
12373 000101F6 AC <1> lodsb
12374 000101F7 FECF <1> dec ch
12375 000101F9 7445 <1> jz short get_argp3
12376 <1> ;push ax
12377 <1> ; 23/07/2022
12378 000101FB 50 <1> push eax
12379 000101FC EBF8 <1> jmp short get_argp1
12380 <1> get_argp_err:
12381 000101FE A3[C08E0100] <1> mov [u.error], eax
12382 <1> ; 14/11/2017
12383 00010203 B801000000 <1> mov eax, ERR_BAD_CMD_ARG ; 01h ; TRDOS 8086
12384 00010208 A3[548E0100] <1> mov [u.r0], eax
12385 0001020D E94CC8FFFF <1> jmp error
12386 <1> get_argp2:
12387 <1> ; 21/07/2015
12388 <1> ;mov eax, 4
12389 00010212 8B15[488F0100] <1> mov edx, [nbase] ; 18/10/2015
12390 00010218 0105[488F0100] <1> add [nbase], eax
12391 <1> ;sub [ncount], ax
12392 <1> ; 23/07/2022
12393 0001021E 2905[4C8F0100] <1> sub [ncount], eax
12394 <1> ;
12395 00010224 8B02 <1> mov eax, [edx]
12396 00010226 C3 <1> retn
12397 <1> get_argpk:
12398 <1> ; Argument is in kernel's memory space
12399 00010227 66C705[4C8F0100]00- <1> mov word [ncount], PAGE_SIZE ; 4096
12400 0001022F 10 <1>
12401 00010230 8935[488F0100] <1> mov [nbase], esi
12402 00010236 8305[488F0100]04 <1> add dword [nbase], 4
12403 0001023D 8B06 <1> mov eax, [esi] ; virtual addr. = physical addr.
12404 0001023F C3 <1> retn
12405 <1> get_argp3:
12406 <1> mov cl, 3
12407 00010242 C1E008 <1> get_argp4:
12408 <1> shl eax, 8
12409 <1> ;pop dx
12410 00010245 5A <1> ; 23/07/2022
12411 00010246 88D0 <1> pop edx
12412 00010248 E2F8 <1> mov al, dl
12413 <1> loop get_argp4
12414 0001024A C3 <1> ;pop esi
12415 <1> retn
12416 <1> ; 23/07/2022
12417 <1> %if 0
12418 <1>
12419 <1> sysstat:
12420 <1> ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
12421 <1> ; temporary !
12422 <1> mov eax, ERR_INV_FNUMBER ; 'invalid function number !'
12423 <1> mov [u.error], eax
12424 <1> mov [u.r0], eax
12425 <1> jmp error
12426 <1>
12427 <1> sysfstat:
12428 <1> ; 13/01/2017 - TRDOS 386 (TRDOS v2.0)
12429 <1> ; temporary !
12430 <1> mov eax, ERR_INV_FNUMBER ; 'invalid function number !'
12431 <1> mov [u.error], eax
12432 <1> mov [u.r0], eax
12433 <1> jmp error
12434 <1>
12435 <1> %endif
12436 <1>
12437 <1> fclose:
12438 <1> ; 24/04/2025 - TRDOS 386 Kernel v2.0.10
12439 <1> ; 18/09/2024 - TRDOS 386 Kernel v2.0.9
12440 <1> ; 23/07/2022 - TRDOS 386 Kernel v2.0.5
12441 <1> ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
12442 <1> ;
12443 <1> ; 18/06/2015 (Retro UNIX 386 v1 - Beginning)
12444 <1> ; (32 bit offset pointer modification)
12445 <1> ; 19/04/2013 - 12/01/2014 (Retro UNIX 8086 v1)
12446 <1> ;
12447 <1> ; Given the file descriptor (index to the u.fp list)
12448 <1> ; 'fclose' first gets the i-number of the file via 'getf'.
12449 <1> ; If i-node is active (i-number > 0) the entry in
12450 <1> ; u.fp list is cleared. If all the processes that opened
12451 <1> ; that file close it, then fsp entry is freed and the file
12452 <1> ; is closed. If not a return is taken.
12453 <1> ; If the file has been deleted while open, 'anyi' is called
12454 <1> ; to see anyone else has it open, i.e., see if it appears
12455 <1> ; in another entry in the fsp table. Upon return from 'anyi'
12456 <1> ; a check is made to see if the file is special.
12457 <1> ;
12458 <1> ; INPUTS ->
12459 <1> ; r1 - contains the file descriptor (value=0,1,2...)
12460 <1> ; u.fp - list of entries in the fsp table
12461 <1> ; fsp - table of entries (4 words/entry) of open files.
12462 <1> ; OUTPUTS ->
12463 <1> ; r1 - contains the same file descriptor
12464 <1> ; r2 - contains i-number
12465 <1> ;
12466 <1> ; ((AX = R1))
12467 <1> ; ((Modified registers: EDX, EBX, ECX, ESI, EDI, EBP))
12468 <1> ;
12469 <1> ; Retro UNIX 8086 v1 modification : CF = 1
12470 <1> ; if i-number of the file is 0. (error)
12471 <1> ;
12472 <1> ; TRDOS 386 (06/10/2016)
12473 <1> ;
12474 <1> ; INPUT:
12475 <1> ; EAX = File Handle (File Descriptor, File Index)
12476 <1> ;
12477 <1> ; OUTPUT:
12478 <1> ; CF = 1 -> File not open !
12479 <1> ; CF = 0 -> OK!
12480 <1> ; EBX = File Number (System)
12481 <1> ; [cdev] = Logical DOS Drive Number
12482 <1> ; EAX = File Handle/Number (user)

```

```

12483 <1> ;
12484 <1> ; Modified Registers: EBX
12485 <1>
12486 0001024B 50 <1> push eax ; File handle
12487 <1>
12488 0001024C E83D000000 <1> call getf
12489 <1> ;jc device_close ; eax = device number
12490 <1> ; 17/04/2021 (temporary)
12491 00010251 7302 <1> jnc short _fclose_0
12492 <1> ; 24/04/2025 (BugFix for 'sysexit')
12493 00010253 58 <1> pop eax
12494 <1> ;jmp rw2 ; file not open !
12495 00010254 C3 <1> retn
12496 <1> _fclose_0:
12497 00010255 80BB[04840100]01 <1> cmp byte [ebx+OF_MODE], 1 ; open mode ; 0 = empty entry
12498 0001025C 7227 <1> jb short fclose_1 ; 1 = read, 2 = write
12499 <1>
12500 <1> ; 18/09/2024 (EMPTY FILE BugFix)
12501 <1> ;cmp eax, 1 ; is the first cluster number > 0
12502 <1> ;jb short fclose_1 ; no, this is empty entry
12503 <1>
12504 <1> fclose_0:
12505 0001025E FE8B[44840100] <1> dec byte [ebx+OF_OPENCOUNT] ; decrement the number of processes
12506 <1> ; that have opened the file
12507 00010264 791F <1> jns short fclose_1 ; jump if not negative (jump if bit 7 is 0)
12508 <1> ; if all processes haven't closed the file, return
12509 <1>
12510 <1> ; eax ; First cluster
12511 00010266 31C0 <1> xor eax, eax ; 0
12512 00010268 8883[04840100] <1> mov [ebx+OF_MODE], al ; 0 = empty entry
12513 <1> ;mov [ebx+OF_STATUS], al ; 0 = empty entry
12514 <1> ;shl bx, 2
12515 <1> ; 23/07/2022
12516 <1> ;shl bl, 2
12517 0001026E C1E302 <1> shl ebx, 2
12518 00010271 8983[64830100] <1> mov [ebx+OF_FCLUSTER], eax ; 0
12519 00010277 8983[E4860100] <1> mov [ebx+OF_CCLUSTER], eax ; 0
12520 <1> ;mov [ebx+OF_CCINDEX], eax ; 0
12521 <1> ; 23/07/2022
12522 <1> ;mov [ebx+OF_OPENCOUNT], al ; 0
12523 0001027D A3[688E0100] <1> mov [u.fofp], eax ; 0
12524 <1> ;shr bx, 2
12525 <1> ; 23/07/2022
12526 <1> ;shr bl, 2
12527 00010282 C1EB02 <1> shr ebx, 2
12528 <1> fclose_1: ; 1:
12529 00010285 58 <1> pop eax ; File handle (File Descriptor, File Index)
12530 00010286 C680[5E8E0100]00 <1> mov byte [eax+u.fp], 0 ; clear that entry in the u.fp list
12531 0001028D C3 <1> retn
12532 <1>
12533 <1> getf:
12534 <1> ; 03/09/2024 - TRDOS 386 v2.0.9
12535 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
12536 <1> ; 17/04/2021 - TRDOS 386 v2.0.4
12537 <1> ; (temporary modifications)
12538 <1> ; 12/10/2016
12539 <1> ; 11/10/2016
12540 <1> ; 08/10/2016
12541 <1> ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
12542 <1> ; / get the device number and the i-number of an open file
12543 <1> ; 13/05/2015
12544 <1> ; 11/05/2015 (Retro UNIX 386 v1 - Beginning)
12545 <1> ; 19/04/2013 - 18/11/2013 (Retro UNIX 8086 v1)
12546 <1> ;
12547 0001028E 89C3 <1> mov ebx, eax
12548 <1> getf1:
12549 00010290 83FB0A <1> cmp ebx, 10
12550 00010293 730A <1> jnb short getf2
12551 00010295 8A9B[5E8E0100] <1> mov bl, [ebx+u.fp]
12552 0001029B 08DB <1> or bl, bl
12553 0001029D 7504 <1> jnz short getf3
12554 <1> getf2:
12555 <1> ; 'File not open !' error (ax=0)
12556 0001029F 29C0 <1> sub eax, eax
12557 <1> ; 03/09/2024
12558 000102A1 F9 <1> stc
12559 000102A2 C3 <1> retn
12560 <1> getf3:
12561 <1> ; 23/07/2022
12562 <1> ;test bl, 80h
12563 <1> ;jnz short getf5 ; device
12564 000102A3 FECB <1> dec bl ; 0 based
12565 000102A5 8A83[E4830100] <1> mov al, [ebx+OF_DRIVE]
12566 000102AB A2[398E0100] <1> mov [cdev], al
12567 000102B0 C0E302 <1> shl bl, 2 ; *4 (dword offset)
12568 <1> ; 23/07/2022
12569 <1> ;shl ebx, 2
12570 000102B3 8B83[E4840100] <1> mov eax, [ebx+OF_SIZE]
12571 000102B9 A3[608F0100] <1> mov [i.size], eax ; file size
12572 000102BE 8D83[64840100] <1> lea eax, [ebx+OF_POINTER] ; 12/10/2016
12573 000102C4 A3[688E0100] <1> mov [u.fofp], eax
12574 000102C9 8B83[64830100] <1> mov eax, [ebx+OF_FCLUSTER]
12575 000102CF C0EB02 <1> shr bl, 2 ; /4 (byte offset)
12576 <1> ; 23/07/2022
12577 <1> ;shr ebx, 2
12578 <1> ; 17/04/2021
12579 000102D2 F8 <1> clc
12580 <1> getf4:
12581 000102D3 C3 <1> retn
12582 <1> ;getf5:
12583 <1> ; 17/04/2021
12584 <1> ; (following code is disabled as temporary)
12585 <1> ;
12586 <1> ; ; get device number
12587 <1> ;and bl, 7Fh ; 1 to 7Fh
12588 <1> ;dec bl ; 0 based (0 to 7Eh)
12589 <1> ;mov al, [ebx+DEV_DRIVER]
12590 <1> ;mov ch, [ebx+DEV_ACCESS]
12591 <1> ;mov cl, [ebx+DEV_OPENMODE]
12592 <1> ;and ch, 0FEh ; reset bit 0 ; dev_close
12593 <1> ;
12594 <1> ; 23/07/2022
12595 <1> ;stc ; cf = 1
12596 <1> ;retn
12597 <1>
12598 <1> trans_addr_nmbp:
12599 <1> ; 18/10/2015
12600 <1> ; 12/10/2015
12601 000102D4 8B2D[708E0100] <1> mov ebp, [u.namep]
12602 <1> trans_addr_nm:
12603 <1> ; Convert virtual (pathname) address to physical address
12604 <1> ; (Retro UNIX 386 v1 feature only !)
12605 <1> ; 18/10/2015
12606 <1> ; 12/10/2015 (u.pnbase & u.pncount has been removed from code)

```

```

12607 <1> ; 02/07/2015
12608 <1> ; 17/06/2015
12609 <1> ; 16/06/2015
12610 <1> ;
12611 <1> ; INPUTS:
12612 <1> ; ebp = pathname address (virtual) ; [u.namep]
12613 <1> ; [u.pgdir] = user's page directory
12614 <1> ; OUTPUT:
12615 <1> ; esi = physical address of the pathname
12616 <1> ; ecx = remain byte count in the page
12617 <1> ;
12618 <1> ; (Modified registers: EAX, EBX, ECX, EDX, ESI)
12619 <1> ;
12620 000102DA 833D[B08E0100]00 <1> ; cmp dword [u.pgdir], 0 ; /etc/init ? (sysexec)
12621 000102E1 7618 <1> jna short trans_addr_nmk ; the caller is os kernel;
12622 <1> ; it is already physical address
12623 000102E3 50 <1> push eax
12624 000102E4 89EB <1> mov ebx, ebp ; [u.namep] ; pathname address (virtual)
12625 000102E6 E81959FFFF <1> call get_physical_addr ; get physical address
12626 000102EB 7204 <1> jc short tr_addr_nm_err
12627 <1> ; 18/10/2015
12628 <1> ; eax = physical address
12629 <1> ; cx = remain byte count in page (1-4096)
12630 <1> ; 12/10/2015 (cx = [u.pncount])
12631 000102ED 89C6 <1> mov esi, eax ; 12/10/2015 (esi=[u.pnbase])
12632 000102EF 58 <1> pop eax
12633 000102F0 C3 <1> retn
12634 <1>
12635 <1> tr_addr_nm_err:
12636 000102F1 A3[C08E0100] <1> mov [u.error], eax
12637 <1> ;pop eax
12638 000102F6 E963C7FFFF <1> jmp error
12639 <1>
12640 <1> trans_addr_nmk:
12641 <1> ; 12/10/2015
12642 <1> ; 02/07/2015
12643 000102FB 8B35[708E0100] <1> mov esi, [u.namep] ; [u.pnbase]
12644 00010301 66B90010 <1> mov cx, PAGE_SIZE ; 4096 ; [u.pncount]
12645 00010305 C3 <1> retn
12646 <1>
12647 <1> sysbreak:
12648 <1> ; 06/09/2024 - TRDOS 386 v2.0.9
12649 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
12650 <1> ; 18/10/2015
12651 <1> ; 07/10/2015
12652 <1> ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
12653 <1> ; 20/06/2013 - 24/03/2014 (Retro UNIX 8086 v1)
12654 <1> ;
12655 <1> ; 'sysbreak' sets the programs break points.
12656 <1> ; It checks the current break point (u.break) to see if it is
12657 <1> ; between "core" and the stack (sp). If it is, it is made an
12658 <1> ; even address (if it was odd) and the area between u.break
12659 <1> ; and the stack is cleared. The new breakpoint is then put
12660 <1> ; in u.break and control is passed to 'sysret'.
12661 <1> ;
12662 <1> ; Calling sequence:
12663 <1> ; sysbreak; addr
12664 <1> ; Arguments: -
12665 <1> ;
12666 <1> ; Inputs: u.break - current breakpoint
12667 <1> ; Outputs: u.break - new breakpoint
12668 <1> ; area between old u.break and the stack (sp) is cleared.
12669 <1> ;
12670 <1> ; .....
12671 <1> ; 06/09/2024 - TRDOS 386 v2.0.9 - Major Modification -
12672 <1> ;
12673 <1> ; This system call performs "malloc" function as in C.
12674 <1> ; Input:
12675 <1> ; EBX = new u.break address (virtual, user's space)
12676 <1> ; (Note: Default/Initial u.break address is the start of the BSS
12677 <1> ; section. Or it is file size for PRG -flat image- files.)
12678 <1> ; If EBX = 0FFFFFFFh ; -1
12679 <1> ; Return current u.break in EAX
12680 <1> ; Output:
12681 <1> ; If EBX input is -1, EAX = current u.break address (virtual)
12682 <1> ; otherwise, EAX = new u.break address
12683 <1> ; Area between the new u.break and the old u.break is cleared.
12684 <1> ; (Note: If the new break address cross overs user's esp,
12685 <1> ; this area <new break - old break> is not cleared.)
12686 <1> ; If CF=1 at return, it means Memory Allocation Error.
12687 <1> ; (or "Insufficient Memory" error)
12688 <1> ; EAX = 0
12689 <1> ; .....
12690 <1> ;
12691 <1> ; Retro UNIX 8086 v1 modification:
12692 <1> ; The user/application program puts breakpoint address
12693 <1> ; in BX register as 'sysbreak' system call argument.
12694 <1> ; (argument transfer method 1)
12695 <1> ;
12696 <1> ; NOTE: Beginning of core is 0 in Retro UNIX 8086 v1 !
12697 <1> ; (!! 'sysbreak' is not needed in Retro UNIX 8086 v1!!)
12698 <1> ;
12699 <1> ; NOTE:
12700 <1> ; 'sysbreak' clears extended part (beyond of previous
12701 <1> ; 'u.break' address) of user's memory for original unix's
12702 <1> ; 'bss' compatibility with Retro UNIX 8086 v1 (19/11/2013)
12703 <1> ;
12704 <1> ; ; mov u.break,r1 / move users break point to r1
12705 <1> ; ; cmp r1,$core / is it the same or lower than core?
12706 <1> ; ; blos lf / yes, lf
12707 00010306 8B2D[848E0100] <1> ; 23/06/2015
12708 <1> mov ebp, [u.break] ; virtual address (offset)
12709 <1> ;
12710 0001030C 31C0 <1> ; 06/09/2024
12711 0001030E A3[548E0100] <1> xor eax, eax
12712 00010313 48 <1> mov [u.r0], eax ; 0 ; default ('memory allocation error')
12713 <1> dec eax ; -1
12714 <1> ;
12715 <1> ; 06/09/2024 - TRDOS 386 v2.0.9
12716 <1> ; get u.break address (for 'malloc' in c compiler)
12717 <1> ; for PRG files:
12718 00010314 39C3 <1> ; default/initial u.break address is file size
12719 00010316 7214 <1> cmp ebx, eax ; -1 ; 0FFFFFFFh
12720 <1> jb short sysbreak_@
12721 <1>
12722 00010318 89EB <1> sysbreak_4:
12723 <1> mov ebx, ebp
12724 0001031A E8E558FFFF <1> ;(allocates a new page for user if it is not present)
12725 0001031F 72D0 <1> call get_physical_addr ; get physical address
12726 <1> jc short tr_addr_nm_err
12727 00010321 892D[548E0100] <1> mov [u.r0], ebp ; start of bss space (to be allocated)
12728 00010327 E952C7FFFF <1> jmp sysret
12729 <1>
12730 <1> sysbreak_@:

```

```

12731 <1> ;and ebp, ebp
12732 <1> ;jz short sysbreak_3
12733 <1> ; Retro UNIX 386 v1 NOTE: u.break points to virtual address !!!
12734 <1> ; (Even break point address is not needed for Retro UNIX 386 v1)
12735 0001032C 8B15[4C8E0100] <1> mov edx, [u.sp] ; kernel stack at the beginning of sys call
12736 00010332 83C20C <1> add edx, 12 ; EIP -4-> CS -4-> EFLAGS -4-> ESP (user)
12737 <1> ; 07/10/2015
12738 <1> ;mov [u.break], ebx ; virtual address !!!
12739 <1> ;
12740 <1> ; 06/09/2024
12741 <1> ; ebp = old/current u.break
12742 <1>
12743 00010335 3B1A <1> cmp ebx, [edx] ; compare new break point with
12744 <1> ; with top of user's stack (virtual!)
12745 00010337 732B <1> jnb short sysbreak_3
12746 <1> ; cmp r1,sp / is it the same or higher
12747 <1> ; / than the stack?
12748 <1> ; bhis 1f / yes, 1f
12749 00010339 89DE <1> mov esi, ebx
12750 0001033B 29EE <1> sub esi, ebp ; new break point - old break point
12751 <1> ;jna short sysbreak_3
12752 <1> ; 06/09/2024
12753 0001033D 7706 <1> ja short sysbreak_1
12754 0001033F 74D7 <1> jz short sysbreak_4
12755 00010341 F7DE <1> neg esi ; convert to positive number (big-small)
12756 00010343 89DD <1> mov ebp, ebx ; move small number (address) to ebp
12757 <1>
12758 <1> sysbreak_1:
12759 <1> ; 06/09/2024
12760 00010345 53 <1> push ebx
12761 00010346 89EB <1> mov ebx, ebp
12762 00010348 E8B758FFFF <1> call get_physical_addr ; get physical address
12763 0001034D 5B <1> pop ebx
12764 0001034E 72A1 <1> jc short tr_addr_nm_err ; 23/07/2022
12765 <1> ; 18/10/2015
12766 00010350 89C7 <1> mov edi, eax
12767 00010352 29C0 <1> sub eax, eax ; 0
12768 <1> ; ECX = remain byte count in page (1-4096)
12769 00010354 39CE <1> cmp esi, ecx
12770 00010356 7302 <1> jnb short sysbreak_2
12771 00010358 89F1 <1> mov ecx, esi
12772 <1> sysbreak_2:
12773 <1> sub esi, ecx
12774 0001035C 01CD <1> add ebp, ecx
12775 0001035E F3AA <1> rep stosb
12776 00010360 09F6 <1> or esi, esi
12777 00010362 75E1 <1> jnz short sysbreak_1
12778 <1> ; bit $1,r1 / is it an odd address
12779 <1> ; beq 2f / no, its even
12780 <1> ; clrb (r1)+ / yes, make it even
12781 <1> ; 2: / clear area between the break point and the stack
12782 <1> ; cmp r1,sp / is it higher or same than the stack
12783 <1> ; bhis 1f / yes, quit
12784 <1> ; clr (r1)+ / clear word
12785 <1> ; br 2b / go back
12786 <1> ;pop ebx
12787 <1> sysbreak_3: ; 1:
12788 <1> ; 06/09/2024
12789 00010364 891D[848E0100] <1> mov [u.break], ebx ; virtual address !!!
12790 <1> ; jsr r0,arg; u.break / put the "address"
12791 <1> ; / in u.break (set new break point)
12792 <1> ; br sysret4 / br sysret
12793 <1> ; 06/09/2024
12794 0001036A 891D[548E0100] <1> mov [u.r0], ebx ; return new break point in eax
12795 00010370 E909C7FFFF <1> jmp sysret
12796 <1>
12797 <1> sysseek: ; / moves read write pointer in an fsp entry
12798 <1> ; 27/12/2025 - TRDOS 386 v2.0.10
12799 <1> ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)
12800 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
12801 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
12802 <1>
12803 <1> ; 'sysseek' changes the r/w pointer of (3rd word of in an
12804 <1> ; fsp entry) of an open file whose file descriptor is in u.r0.
12805 <1> ; The file descriptor refers to a file open for reading or
12806 <1> ; writing. The read (or write) pointer is set as follows:
12807 <1> ; * if 'ptrname' is 0, the pointer is set to offset.
12808 <1> ; * if 'ptrname' is 1, the pointer is set to its
12809 <1> ; current location plus offset.
12810 <1> ; * if 'ptrname' is 2, the pointer is set to the
12811 <1> ; size of file plus offset.
12812 <1> ; The error bit (e-bit) is set for an undefined descriptor.
12813 <1>
12814 <1> ; Calling sequence:
12815 <1> ; sysseek; offset; ptrname
12816 <1> ; Arguments:
12817 <1> ; offset - number of bytes desired to move
12818 <1> ; the r/w pointer
12819 <1> ; ptrname - a switch indicated above
12820 <1>
12821 <1> ; Inputs: r0 - file descriptor
12822 <1> ; 27/12/2025
12823 <1> ; EBX = File descriptor (number) -user-
12824 <1> ; ECX = Offset (plus)
12825 <1> ; EDX = 0 -> set to offset in ECX
12826 <1> ; = 1 -> set to current offset + offset in ECX
12827 <1> ; = 2 -> set to file size + offset in ECX
12828 <1> ; Outputs: -
12829 <1> ; EAX = (new) File Offset ; 27/12/2025
12830 <1> ; .....
12831 <1>
12832 <1> ; Retro UNIX 8086 v1 modification:
12833 <1> ; 'sysseek' system call has three arguments; so,
12834 <1> ; * 1st argument, file descriptor is in BX (BL) register
12835 <1> ; * 2nd argument, offset is in CX register
12836 <1> ; * 3rd argument, ptrname/switch is in DX (DL) register
12837 <1>
12838 00010375 E823000000 <1> call seektell
12839 <1> ; EAX = Current R/W pointer of the file
12840 <1> ; EBX = [u.fofp]
12841 <1> ; [u.base] = offset (ECX input)
12842 <1>
12843 0001037A 0305[788E0100] <1> add eax, [u.base]
12844 00010380 8903 <1> mov [ebx], eax
12845 <1> ; 27/12/2025
12846 <1> sysseek_@:
12847 00010382 A3[548E0100] <1> mov [u.r0], eax ; return (new) file offset
12848 00010387 E9F2C6FFFF <1> jmp sysret
12849 <1>
12850 <1> systell: ; / get the r/w pointer
12851 <1> ; 27/12/2025 - TRDOS 386 v2.0.10
12852 <1> ; 06/11/2016 - TRDOS 386 (TRDOS v2.0) - temporary !-
12853 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
12854 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)

```

```

12855 <1> ;
12856 <1> ; Retro UNIX 8086 v1 modification:
12857 <1> ; ! 'system' does not work in original UNIX v1,
12858 <1> ; it returns with error !
12859 <1> ; Inputs: r0 - file descriptor
12860 <1> ; 27/12/2025
12861 <1> ; EBX = File descriptor (number) -user-
12862 <1> ; ECX = ignored
12863 <1> ; EDX = 0 -> return current file offset
12864 <1> ; = 1 -> return current file offset
12865 <1> ; = 2 -> return file size
12866 <1> ; > 2 -> return current file offset
12867 <1> ;
12868 <1> ; Outputs: r0 - file r/w pointer
12869 <1>
12870 <1> ;xor ecx, ecx ; 0
12871 <1>
12872 <1> ; 27/12/2025
12873 0001038C 83FA02 <1> cmp edx, 2
12874 0001038F 7405 <1> je short systell_@
12875 00010391 BA01000000 <1> mov edx, 1 ; 05/08/2013
12876 <1> systell_@:
12877 <1> ;call seektell
12878 00010396 E808000000 <1> call seektell0 ; 05/08/2013
12879 <1> ; 06/11/2016
12880 <1> ;mov eax, [ebx]
12881 <1> ; 27/12/2025
12882 <1> ;mov [u.r0], eax
12883 <1> ;jmp sysret
12884 0001039B EBE5 <1> jmp short sysseek_@
12885 <1>
12886 <1> ; Original unix v1 'system' system call:
12887 <1> ; jsr r0,seektell
12888 <1> ; br error4
12889 <1>
12890 <1> seektell:
12891 <1> ; 24/04/2025 - TRDOS 386 v2.0.10
12892 <1> ; 06/11/2016 - TRDOS 386 (TRDOS v2.0)
12893 <1> ; 03/01/2016
12894 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
12895 <1> ; 07/07/2013 - 05/08/2013 (Retro UNIX 8086 v1)
12896 <1>
12897 <1> ; 'seektell' puts the arguments from sysseek and systell
12898 <1> ; call in u.base and u.count. It then gets the i-number of
12899 <1> ; the file from the file descriptor in u.r0 and by calling
12900 <1> ; getf. The i-node is brought into core and then u.count
12901 <1> ; is checked to see it is a 0, 1, or 2.
12902 <1> ; If it is 0 - u.count stays the same
12903 <1> ; 1 - u.count = offset (u.fofp)
12904 <1> ; 2 - u.count = i.size (size of file)
12905 <1>
12906 <1> ; !! Retro UNIX 8086 v1 modification:
12907 <1> ; Argument 1, file descriptor is in BX;
12908 <1> ; Argument 2, offset is in CX;
12909 <1> ; Argument 3, ptrname/switch is in DX register.
12910 <1>
12911 <1> ; ((Return -> eax = base for offset (position = base+offset))
12912 <1> ;
12913 0001039D 890D[788E0100] <1> mov [u.base], ecx ; offset
12914 <1> seektell0:
12915 000103A3 8915[7C8E0100] <1> mov [u.count], edx
12916 <1> ; EBX = file descriptor (file number)
12917 000103A9 E8E2FEFFFF <1> call getf1
12918 <1> ; EAX = First cluster of the file
12919 <1> ; EBX = File number (Open file number)
12920 <1> ; [u.fofp] = Pointer to File pointer
12921 <1> ; [i.size] = File size
12922 <1>
12923 000103AE 09C0 <1> or eax, eax
12924 000103B0 7505 <1> jnz short seektell1
12925 <1>
12926 <1> ; 24/04/2025 - TRDOS 386 v2.0.10
12927 <1> ;mov eax, ERR_FILE_NOT_OPEN
12928 <1> ;mov [u.r0], eax
12929 <1> ;mov dword [u.error], eax ; 'file not open !'
12930 <1> ;jmp error
12931 <1> ; 24/04/2025
12932 000103B2 E9E6CDFFFF <1> jmp seektell_err ; sysclose_err
12933 <1>
12934 <1> seektell1:
12935 000103B7 8B1D[688E0100] <1> mov ebx, [u.fofp]
12936 000103BD 803D[7C8E0100]01 <1> cmp byte [u.count], 1
12937 000103C4 7705 <1> ja short seektell2
12938 000103C6 7409 <1> je short seektell3
12939 000103C8 31C0 <1> xor eax, eax
12940 000103CA C3 <1> retn
12941 <1>
12942 <1> seektell2:
12943 000103CB A1[608F0100] <1> mov eax, [i.size]
12944 000103D0 C3 <1> retn
12945 <1>
12946 <1> seektell3:
12947 000103D1 8B03 <1> mov eax, [ebx]
12948 000103D3 C3 <1> retn
12949 <1>
12950 <1> sysintr: ; / set interrupt handling
12951 <1> ; 21/08/2024 - TRDOS 386 v2.0.9 modification
12952 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
12953 <1> ; 07/07/2013 (Retro UNIX 8086 v1)
12954 <1>
12955 <1> ; 'sysintr' sets the interrupt handling value. It puts
12956 <1> ; argument of its call in u.intr then branches into 'sysquit'
12957 <1> ; routine. u.tty is checked if to see if a control tty exists.
12958 <1> ; If one does the interrupt character in the tty buffer is
12959 <1> ; cleared and 'sysret' is called. If one does not exits
12960 <1> ; 'sysret' is just called.
12961 <1>
12962 <1> ; Calling sequence:
12963 <1> ; sysintr; arg
12964 <1> ; Argument:
12965 <1> ; arg - if 0, interrupts (ASCII DELETE) are ignored.
12966 <1> ; - if 1, interrupts cause their normal result
12967 <1> ; i.e force an exit.
12968 <1> ; - if arg is a location within the program,
12969 <1> ; control is passed to that location when
12970 <1> ; an interrupt occurs.
12971 <1> ; Inputs: -
12972 <1> ; Outputs: -
12973 <1> ; .....
12974 <1>
12975 <1> ; Retro UNIX 8086 v1 modification:
12976 <1> ; 'sysintr' system call sets u.intr to value of BX
12977 <1> ; then branches into sysquit.
12978 <1>

```



```

12979 <1> ; 21/08/2024 - TRDOS 386
12980 <1> ; enable/disable terminate (CTRL+BREAK) interrupt
12981 <1>
12982 <1> ; INPUT:
12983 <1> ; bx = 0 -> disable CTRL+BREAK
12984 <1> ; -u.intr will be set to 0, u.quit will be ignored-
12985 <1> ; bx > 0 -> enable CTRL+BREAK (also default at start)
12986 <1> ; -u.intr will be set to 0FFFFh, u.quit will be used-
12987 <1> ; NOTE: u.quit is flag for CTRL+BREAK key press status
12988 <1> ; -1 = pressed -termination request-, 0 = not pressed
12989 <1> ; OUTPUT:
12990 <1> ; none
12991 <1>
12992 <1> ;;;
12993 <1> ; 21/08/2024
12994 000103D4 F7C3FFFF0000 <1> test ebx, 0FFFFh
12995 000103DA 7403 <1> jz short sysintr_@ ; 0
12996 <1> ;mov bx, 0FFFFh
12997 000103DC 31DB <1> xor ebx, ebx
12998 000103DE 4B <1> dec ebx ; -1
12999 <1> sysintr_@:
13000 <1> ;;;
13001 000103DF 66891D[A08E0100] <1> mov [u.intr], bx
13002 <1> ; jsr r0,arg; u.intr / put the argument in u.intr
13003 <1> ; br 1f / go into quit routine
13004 000103E6 E993C6FFFF <1> jmp sysret
13005 <1>
13006 <1> sysquit:
13007 <1> ; 21/08/2024 - TRDOS 386 v2.0.9 modification
13008 <1> ; get/reset QUIT (u.quit) status
13009 <1>
13010 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
13011 <1> ; 07/07/2013 (Retro UNIX 8086 v1)
13012 <1>
13013 <1> ; 'sysquit' turns off the quit signal. it puts the argument of
13014 <1> ; the call in u.quit. u.tty is checked if to see if a control
13015 <1> ; tty exists. If one does the interrupt character in the tty
13016 <1> ; buffer is cleared and 'sysret' is called. If one does not exits
13017 <1> ; 'sysret' is just called.
13018 <1>
13019 <1> ; Calling sequence:
13020 <1> ; sysquit; arg
13021 <1> ; Argument:
13022 <1> ; arg - if 0, this call disables quit signals from the
13023 <1> ; typewriter (ASCII FS)
13024 <1> ; - if 1, quits are re-enabled and cause execution to
13025 <1> ; cease and a core image to be produced.
13026 <1> ; i.e force an exit.
13027 <1> ; - if arg is an address in the program,
13028 <1> ; a quit causes control to sent to that
13029 <1> ; location.
13030 <1> ; Inputs: -
13031 <1> ; Outputs: -
13032 <1> ; .....
13033 <1>
13034 <1> ; Retro UNIX 8086 v1 modification:
13035 <1> ; 'sysquit' system call sets u.quit to value of BX
13036 <1> ; then branches into 'sysret'.
13037 <1>
13038 <1> ; 21/08/2024
13039 <1> ;mov [u.quit], bx
13040 <1> ;jmp sysret
13041 <1> ; jsr r0,arg; u.quit / put argument in u.quit
13042 <1> ;1:
13043 <1> ; mov u.ttyp,r1 / move pointer to control tty buffer
13044 <1> ; ; to r1
13045 <1> ; beq sysret4 / return to user
13046 <1> ; clrb 6(r1) / clear the interrupt character
13047 <1> ; ; in the tty buffer
13048 <1> ; br sysret4 / return to user
13049 <1>
13050 <1> ; 21/08/2024 - TRDOS 386
13051 <1> ; get/reset QUIT (u.quit) status
13052 <1>
13053 <1> ; INPUT:
13054 <1> ; none
13055 <1> ; OUTPUT:
13056 <1> ; eax = 0 -> no CTRL+BREAK
13057 <1> ; eax = -1 -> CTRL+BREAK status (but not applied)
13058 <1> ; -u.intr may be 0-
13059 <1> ; NOTE: u.quit will be reset to 0
13060 <1>
13061 000103EB 31C0 <1> xor eax, eax ; 0
13062 000103ED 663905[A28E0100] <1> cmp [u.quit], ax
13063 000103F4 7407 <1> jz short sysquit_@ ; already 0
13064 000103F6 66A3[A28E0100] <1> mov [u.quit], ax ; 0 ; reset
13065 000103FC 4B <1> dec eax ; -1
13066 <1> sysquit_@:
13067 000103FD A3[548E0100] <1> mov [u.r0], eax
13068 00010402 E977C6FFFF <1> jmp sysret
13069 <1>
13070 <1> %if 0
13071 <1>
13072 <1> anyi:
13073 <1> ; 23/07/2022
13074 <1> ; 06/10/2016 (TRDOS 386 = TRDOS v2.0)
13075 <1> ; Major Modification!
13076 <1> ; TRDOS 386 does not permit to delete a file while it is open
13077 <1> ; The role of 'anyi' procedure has been changed to ensure that.
13078 <1>
13079 <1> ; 22/06/2015 (Retro UNIX 386 v1 - Beginning)
13080 <1> ; 25/04/2013 (Retro UNIX 8086 v1)
13081 <1>
13082 <1> ; 'anyi' is called if a file deleted while open.
13083 <1> ; "anyi" checks to see if someone else has opened this file.
13084 <1>
13085 <1> ; INPUTS ->
13086 <1> ; r1 - contains an i-number
13087 <1> ; fsp - start of table containing open files
13088 <1>
13089 <1> ; OUTPUTS ->
13090 <1> ; "deleted" flag set in fsp entry of another occurrence of
13091 <1> ; this file and r2 points 1st word of this fsp entry.
13092 <1> ; if file not found - bit in i-node map is cleared
13093 <1> ; (i-node is freed)
13094 <1> ; all blocks related to i-node are freed
13095 <1> ; all flags in i-node are cleared
13096 <1> ; ((AX = R1)) input
13097 <1>
13098 <1> ; (Retro UNIX Prototype : 02/12/2012, UNIXCOPY.ASM)
13099 <1> ; ; ((Modified registers: EDX, ECX, EBX, ESI, EDI, EBP))
13100 <1>
13101 <1> ; / r1 contains an i-number
13102 <1>

```

```

13103 <1> ; TRDOS 386 (06/10/2016)
13104 <1> ;
13105 <1> ; INPUT:
13106 <1> ; EAX = First Cluster
13107 <1> ; DL = Logical DOS Drive Number
13108 <1> ;
13109 <1> ; OUTPUT:
13110 <1> ; CF = 1 -> EBX = File Handle/Number/Index
13111 <1> ; CF = 0 -> EBX = 0
13112 <1> ;
13113 <1> ; Modified Registers: EBX
13114 <1>
13115 <1> xor ebx, ebx
13116 <1> anyi_0:
13117 <1> cmp byte [ebx+OF_MODE], 0 ; 0 = empty entry
13118 <1> ja short anyi_2 ; 1 (r), 2 (w) or 3 (r&w)
13119 <1> anyi_1:
13120 <1> inc bl
13121 <1> cmp bl, OPENFILES ; max. count of open files
13122 <1> jb short anyi_0
13123 <1> xor eax, eax
13124 <1> retn
13125 <1> anyi_2:
13126 <1> cmp dl, [ebx+OF_DRIVE]
13127 <1> jne short anyi_1
13128 <1> ;shl bx, 2 ; *4 (dword offset)
13129 <1> shl ebx, 2 ; 23/07/2022
13130 <1> cmp eax, [ebx+OF_FCLUSTER]
13131 <1> je short anyi_3
13132 <1> ;shr bx, 2 ; /4 (byte offset)
13133 <1> shr ebx, 2 ; 23/07/2022
13134 <1> jmp short anyi_1
13135 <1> anyi_3:
13136 <1> ;shr bx, 2 ; /4 (bytes offset) (index)
13137 <1> shr ebx, 2 ; 23/07/2022
13138 <1> stc
13139 <1> retn
13140 <1>
13141 <1> %endif
13142 <1>
13143 <1> ; Retro UNIX 386 v1 Kernel (v0.2) - SYS9.INC
13144 <1> ; Last Modification: 09/12/2015
13145 <1>
13146 <1> sysssleep:
13147 <1> ; 28/08/2024 - TRDOS 386 v2.0.9
13148 <1> ; 24/07/2022 - TRDOS 386 v2.0.5
13149 <1> ; 29/06/2015 - (Retro UNIX 386 v1)
13150 <1> ; 11/06/2014 - (Retro UNIX 8086 v1)
13151 <1> ;
13152 <1> ; Retro UNIX 8086 v1 feature only
13153 <1> ; (INPUT -> none)
13154 <1>
13155 <1> ; Temporary - 24/07/2022
13156 00010407 891D[548E0100] <1> mov [u.r0], ebx
13157 <1> ;
13158 0001040D 0FB61D[A58E0100] <1> movzx ebx, byte [u.uno] ; process number
13159 00010414 8AA3[678D0100] <1> mov ah, [ebx+p.ttyc-1] ; current/console tty
13160 0001041A E8B4150000 <1> call sleep
13161 <1>
13162 <1> ; 23/08/2024 - restore [u.usp]
13163 <1> ; swap changes [u.usp] to esp which points to
13164 <1> ; return of sleep
13165 0001041F 8925[508E0100] <1> mov [u.usp], esp
13166 <1> ; points to user's regs on top ofn system stack
13167 <1>
13168 <1> ; 24/07/2022
13169 00010425 FF05[548E0100] <1> inc dword [u.r0] ; Temporary !
13170 0001042B E94EC6FFFF <1> jmp sysret
13171 <1>
13172 <1> _vp_clr:
13173 <1> ; Reset/Clear Video Page
13174 <1> ;
13175 <1> ; 24/07/2022 - TRDOS 386 v2.0.5
13176 <1> ; 30/06/2015 - (Retro UNIX 386 v1)
13177 <1> ; 21/05/2013 - 30/10/2013(Retro UNIX 8086 v1) (U0.ASM)
13178 <1> ;
13179 <1> ; Retro UNIX 8086 v1 feature only !
13180 <1> ;
13181 <1> ; INPUTS ->
13182 <1> ; BH = video page number
13183 <1> ;
13184 <1> ; OUTPUT ->
13185 <1> ; none
13186 <1> ; ((Modified registers: EAX, BH, ECX, EDX, ESI, EDI))
13187 <1> ;
13188 <1> ; 04/12/2013
13189 00010430 28C0 <1> sub al, al
13190 <1> ; al = 0 (clear video page)
13191 <1> ; bh = video page ; 13/05/2016
13192 00010432 B407 <1> mov ah, 07h
13193 <1> ; ah = 7 (attribute/color)
13194 <1> ;xor cx, cx ; 0, left upper column (cl) & row (cl)
13195 <1> ;mov dx, 184Fh ; right lower column & row (dl=24, dh=79)
13196 <1> ; 24/07/2022
13197 00010434 31C9 <1> xor ecx, ecx
13198 00010436 BA4F180000 <1> mov edx, 184Fh
13199 0001043B E8ED18FFFF <1> call _scroll_up
13200 <1> ; bh = video page
13201 <1> ;xor dx, dx ; 0 (cursor position)
13202 <1> ; 24/07/2022
13203 00010440 31D2 <1> xor edx, edx
13204 00010442 E92C1FFFFF <1> jmp _set_cpos
13205 <1>
13206 <1> sysmsg:
13207 <1> ; 28/08/2024 - TRDOS 386 v2.0.9
13208 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
13209 <1> ; 07/12/2020
13210 <1> ; 05/12/2020
13211 <1> ; 13/05/2016
13212 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
13213 <1> ; 01/07/2015 - 11/11/2015 (Retro UNIX 386 v1)
13214 <1> ; Print user-application message on user's console tty
13215 <1> ;
13216 <1> ; Input -> EBX = Message address
13217 <1> ; ECX = Message length (max. 255)
13218 <1> ; DL = Color (IBM PC Rombios color attributes)
13219 <1> ;
13220 00010447 81F9FF000000 <1> cmp ecx, MAX_MSG_LEN ; 255
13221 <1> ;ja sysret ; nothing to do with big message size
13222 <1> ; 23/07/2022
13223 0001044D 7605 <1> jna short sysmsg8
13224 <1> sysmsg7:
13225 0001044F E92AC6FFFF <1> jmp sysret
13226 <1> sysmsg8: ; 23/07/2022

```

```

13227 00010454 08C9      <1>      or      cl, cl
13228                      <1>      ;jz      sysret
13229                      <1>      ; 23/07/2022
13230 00010456 74F7      <1>      jz      short sysmsg7
13231 00010458 20D2      <1>      and     dl, dl
13232 0001045A 7502      <1>      jnz     short sysmsg0
13233 0001045C B207      <1>      mov     dl, 07h ; default color
13234                      <1>      ; (black background, light gray character)
13235                      <1>      sysmsg0:
13236 0001045E 891D[788E0100] <1>      mov     [u.base], ebx
13237 00010464 8815[CF760100] <1>      mov     [ccolor], dl ; color attributes
13238 0001046A 89E5      <1>      mov     ebp, esp ; save stack pointer
13239 0001046C 31DB      <1>      xor     ebx, ebx ; 0
13240 0001046E 891D[808E0100] <1>      mov     [u.nread], ebx ; 0
13241                      <1>      ;
13242 00010474 381D[BA8E0100] <1>      cmp     [u.kcall], bl ; 0
13243 0001047A 7770      <1>      ja      short sysmsgk ; Temporary (01/07/2015)
13244                      <1>      ;
13245 0001047C 890D[7C8E0100] <1>      mov     [u.count], ecx
13246                      <1>      ;inc     ecx ; + 00h ; ASCIIIZ
13247                      <1>      ;
13248                      <1>      ; 07/12/2020
13249                      <1>      ;add     ecx, 3
13250 00010482 6683C103 <1>      add     cx, 3
13251 00010486 80E1FC      <1>      and     cl, ~3 ; not 3
13252                      <1>      ;
13253                      <1>      ;;;
13254                      <1>      ; 20/08/2024
13255                      <1>      ; for safety, against a possibility which if ecx+1
13256                      <1>      ; chars (without a 0 in ecx chars) overs stack frame
13257                      <1>      ; (for example 255 chars without 0, safe stack frame
13258                      <1>      ; size is 260.)
13259                      <1>      ; !!! 'sysmsg' will put a 0 at ecx+1 position.!!!
13260 00010489 51        <1>      push     ecx
13261                      <1>      ;
13262                      <1>      ;;;
13263 0001048A 29CC      <1>      sub     esp, ecx
13264 0001048C 89E7      <1>      mov     edi, esp
13265 0001048E 89E6      <1>      mov     esi, esp
13266 00010490 66891D[B88E0100] <1>      mov     [u.pcount], bx ; reset page (phy. addr.) counter
13267                      <1>      ; 11/11/2015
13268 00010497 8A25[888E0100] <1>      mov     ah, [u.ttyp] ; recent open tty
13269                      <1>      ; 0 = none
13270 0001049D FECC      <1>      dec     ah
13271 0001049F 790C      <1>      jns     short sysmsg1
13272 000104A1 8A1D[A58E0100] <1>      mov     bl, [u.uno] ; process number
13273 000104A7 8AA3[678D0100] <1>      mov     ah, [ebx+p.ttyp-1] ; user's (process's) console tty
13274                      <1>      sysmsg1:
13275 000104AD 8825[8A8E0100] <1>      mov     [u.ttyn], ah
13276                      <1>      sysmsg2:
13277 000104B3 E8F4050000 <1>      call    cpass
13278 000104B8 7416      <1>      jz      short sysmsg5
13279 000104BA AA        <1>      stosb
13280 000104BB 20C0      <1>      and     al, al
13281 000104BD 75F4      <1>      jnz     short sysmsg2
13282                      <1>      sysmsg3:
13283 000104BF 80FC07      <1>      cmp     ah, 7 ; tty number
13284 000104C2 7711      <1>      ja      short sysmsg6 ; serial port
13285 000104C4 E83E000000 <1>      call    print_cmsg ; 05/12/2020
13286                      <1>      sysmsg4:
13287 000104C9 89EC      <1>      mov     esp, ebp ; restore stack pointer
13288 000104CB E9AEC5FFFF <1>      jmp     sysret
13289                      <1>      sysmsg5:
13290 000104D0 C60700      <1>      mov     byte [edi], 0
13291 000104D3 EBFA      <1>      jmp     short sysmsg3
13292                      <1>      sysmsg6:
13293 000104D5 8A06      <1>      mov     al, [esi]
13294 000104D7 E8F7140000 <1>      call    sndc
13295 000104DC 72EB      <1>      jc      short sysmsg4
13296 000104DE 803E00      <1>      cmp     byte [esi], 0 ; 0 is stop character
13297 000104E1 76E6      <1>      jna     short sysmsg4
13298 000104E3 46        <1>      inc     esi
13299 000104E4 8A25[8A8E0100] <1>      mov     ah, [u.ttyn]
13300 000104EA EBE9      <1>      jmp     short sysmsg6
13301                      <1>
13302                      <1>      sysmsgk: ; Temporary (01/07/2015)
13303                      <1>      ; The message has been sent by kernel (ASCIIIZ string)
13304                      <1>      ; (ECX -character count- will not be considered)
13305 000104EC 8B35[788E0100] <1>      mov     esi, [u.base]
13306 000104F2 8A25[CE760100] <1>      mov     ah, [ptty] ; present/current screen (video page)
13307 000104F8 8825[8A8E0100] <1>      mov     [u.ttyn], ah
13308 000104FE C605[BA8E0100]00 <1>      mov     byte [u.kcall], 0
13309 00010505 EBB8      <1>      jmp     short sysmsg3
13310                      <1>
13311                      <1>      print_cmsg:
13312                      <1>      ; 08/12/2020
13313                      <1>      ; 07/12/2020
13314                      <1>      ; 05/12/2020
13315                      <1>      ; 18/11/2017
13316                      <1>      ; 13/05/2016 - TRDOS 386 (TRDOS v2.0)
13317                      <1>      ; 01/07/2015 (Retro UNIX 386 v1)
13318                      <1>      ;
13319                      <1>      ; print message (on user's console tty)
13320                      <1>      ; with requested color
13321                      <1>      ;
13322                      <1>      ; INPUTS:
13323                      <1>      ; esi = message address
13324                      <1>      ; [u.ttyn] = tty number (0 to 7)
13325                      <1>      ; [ccolor] = color attributes (IBM PC BIOS colors)
13326                      <1>      ;
13327                      <1>      ; Modified registers: eax, ebx, ecx, edx, esi, edi
13328                      <1>      ; (ebp must be preserved)
13329                      <1>      ;
13330                      <1>      ;mov     bh, ah
13331 00010507 8A3D[8A8E0100] <1>      mov     bh, [u.ttyn]
13332 0001050D 8A1D[CF760100] <1>      mov     bl, [ccolor] ; * ; 05/12/2020
13333                      <1>      ;
13334                      <1>      ; 05/12/2020
13335 00010513 803D[0C9D0100]00 <1>      cmp     byte [pmi32], 0 ; is vbiOS's 32 bit pmi enabled ?
13336 0001051A 772E      <1>      ja      short pcmsg5 ; yes
13337                      <1>      pcmsg1:
13338                      <1>      ; 08/12/2020
13339 0001051C 8A1D[CF760100] <1>      mov     bl, [ccolor] ; * (video.s 'u11'&'beep' change BL)
13340                      <1>      ;
13341 00010522 AC        <1>      lodsb
13342 00010523 20C0      <1>      and     al, al ; 0
13343 00010525 743A      <1>      jz      short pcmsg2
13344                      <1>      pcmsg7:
13345 00010527 56        <1>      push     esi
13346                      <1>      ;mov     bl, [ccolor] ; * (video.s 'u11'&'beep' change BL)
13347                      <1>      ; 05/12/2020
13348                      <1>      ;mov     bh, [u.ttyn]
13349                      <1>      ;call    _write_tty
13350                      <1>      ;pop     esi

```

```

13351      <1>      ;jmp      short pcmsg1
13352      <1> ;pcmsg2:
13353      <1>      ;retn
13354      <1>
13355      <1>      ; 07/12/2020
13356      <1>      cmp      byte [CRT_MODE], 3
13357      <1>      ja       short pcmsg4
13358      <1> pcmsg3:
13359      <1>      call     _write_tty_m3
13360      <1>      pop      esi
13361      <1>      jmp      short pcmsg1
13362      <1> pcmsg4:
13363      <1>      cmp      byte [CRT_MODE], 7
13364      <1>      jna      short pcmsg3
13365      <1>      call     vga_write_teletype
13366      <1>      pop      esi
13367      <1>      jmp      short pcmsg1
13368      <1> pcmsg5:
13369      <1>      ; 07/12/2020
13370      <1>      cmp      byte [CRT_MODE], 7
13371      <1>      jna      short pcmsg1
13372      <1>
13373      <1>      ; 05/12/2020
13374      <1>      ; writing message by using
13375      <1>      ; VESA VBE3 video bios protected mode interface
13376      <1>
13377      <1>      mov      ah, 0Eh
13378      <1> pcmsg6:
13379      <1>      lodsb
13380      <1>      and      al, al ; 0
13381      <1>      jz       short pcmsg2
13382      <1>      ; bh = video page
13383      <1>      ; ah = 0Eh
13384      <1>      ; al = character
13385      <1>      ; bl = color
13386      <1>      call     int10h_32bit_pmi
13387      <1>      jmp      short pcmsg6
13388      <1> pcmsg2:
13389      <1>      retn
13390      <1>
13391      <1> sysgeterr:
13392      <1>      ; 09/12/2015
13393      <1>      ; 21/09/2015 - (Retro UNIX 386 v1 feature only!)
13394      <1>      ; Get last error number or page fault count
13395      <1>      ; (for debugging)
13396      <1>
13397      <1>      ; Input -> EBX = return type
13398      <1>      ;      0 = last error code (which is in 'u.error')
13399      <1>      ;      FFFFFFFFh = page fault count for running process
13400      <1>      ;      FFFFFFFEh = total page fault count
13401      <1>      ;      1 .. FFFFFFFDh = undefined
13402      <1>
13403      <1>      ; Output -> EAX = last error number or page fault count
13404      <1>      ; (depending on EBX input)
13405      <1>
13406      <1>      and      ebx, ebx
13407      <1>      jnz      short glerr_2
13408      <1> glerr_0:
13409      <1>      mov      eax, [u.error]
13410      <1> glerr_1:
13411      <1>      mov      [u.r0], eax
13412      <1>      retn
13413      <1> glerr_2:
13414      <1>      inc      ebx ; FFFFFFFFh -> 0, FFFFFFFEh -> FFFFFFFFh
13415      <1>      jz       short glerr_2 ; page fault count for process
13416      <1>      inc      ebx ; FFFFFFFFh -> 0
13417      <1>      jnz      short glerr_0
13418      <1>      mov      eax, [PF_Count] ; total page fault count
13419      <1>      jmp      short glerr_1
13420      <1> glerr_3:
13421      <1>      mov      eax, [u.pfcount]
13422      <1>      jmp      short glerr_1
13423      <1>
13424      <1> load_and_run_file:
13425      <1>      ; 10/01/2026
13426      <1>      ; 07/07/2025 - TRDOS 386 Kernel v2.0.10
13427      <1>      ; 23/07/2022 - TRDOS 386 Kernel v2.0.5
13428      <1>      ; 18/11/2017
13429      <1>      ; 22/01/2017
13430      <1>      ; 04/01/2017 - 07/01/2017
13431      <1>      ; 24/10/2016
13432      <1>      ; 24/04/2016 - 02/05/2016 - 03/05/2016 - 06/05/2016
13433      <1>      ; 23/04/2016 (TRDOS 386 = TRDOS v2.0)
13434      <1>      ; 23/10/2015 (Retro UNIX 386 v1, 'sysexec')
13435      <1>      ; 23/06/2015 (Retro UNIX 386 v1 - Beginning)
13436      <1>      ; 03/06/2013 - 06/12/2013 (Retro UNIX 8086 v1)
13437      <1>      ; EAX = First Cluster number
13438      <1>      ; EDX = File Size
13439      <1>      ; ESI = Argument list address
13440      <1>      ; [argc] = argument count
13441      <1>      ; [u.nread] = argument list length
13442      <1>      ; [esp] = return address to the caller (*)
13443      <1>
13444      <1>      mov      [argv], esi
13445      <1>      mov      [i.size], edx
13446      <1>      mov      [ii], eax
13447      <1>
13448      <1>      ; sti ; 07/01/2017
13449      <1>      ; mov      eax, [k_page_dir]
13450      <1>      ; mov      [u.pgdir], eax
13451      <1>      xor      eax, eax ; cll ; *** ; 04/01/2017
13452      <1>      ; mov      [u.r0], eax ; 0 ; 07/01/2017
13453      <1>
13454      <1>      ; 06/05/2016
13455      <1>      ; Set 'sysexit' return order to MainProg
13456      <1>
13457      <1>      pop      eax ; * 'loc_load_and_run_file_8:' address
13458      <1>      ; 22/01/2017
13459      <1>      ; cll ; 07/01/2017
13460      <1>      mov      esp, [tss.esp0]
13461      <1>
13462      <1>      ; 'loc_load_run_file_8' address has
13463      <1>      ; 'jmp loc_file_rw_restore_retn' instruction
13464      <1>      ; 'loc_file_rw_restore_retn:' will return to
13465      <1>      ; [mainprog_return_addr]
13466      <1>      ; just after 'call command_interpreter'
13467      <1>
13468      <1>      push     _end_of_mainprog ; we must not return to here !
13469      <1>      push     dword [mainprog_return_addr]
13470      <1>      mov      ebp, esp ; **
13471      <1>
13472      <1>      pushfd   ; EFLAGS ; IRETD ; ***
13473      <1>      push     KC0DE ; cs ; IRETD
13474      <1>      push     eax ; * (eip) ; IRETD

```

```

13475 000105B0 8925[4C8E0100] <1> mov [u.sp], esp
13476 <1> ;mov byte [u.quant], time_count
13477 000105B6 1E <1> push ds
13478 000105B7 06 <1> push es
13479 000105B8 0FA0 <1> push fs
13480 000105BA 0FA8 <1> push gs
13481 <1> ;mov eax, [u.r0]
13482 000105BC 29C0 <1> sub eax, eax
13483 000105BE 60 <1> pushad
13484 000105BF 68[7ECA0000] <1> push sysret
13485 <1> ;push sysret11 ; 07/01/2017
13486 000105C4 8925[508E0100] <1> mov [u.usp], esp
13487 <1> ;
13488 000105CA E8F8030000 <1> call wswap ; Save MainProg (process 1) 'u' structure
13489 <1> ; and registers for return (from program)
13490 000105CF 89EC <1> mov esp, ebp ; **
13491 <1> ;22/01/2017
13492 <1> ;sti ; 07/01/2017
13493 <1> ; 23/07/2022
13494 <1> ;push eax ; * 'loc_load_and_run_file_8:' address
13495 <1> ;
13496 <1> ;;; 02/05/2016
13497 <1> ;;; Create a new process (parent: MainProg)
13498 000105D1 31F6 <1> xor esi, esi
13499 <1> cnpm_1: ; search p.stat table for unused process number
13500 000105D3 46 <1> inc esi
13501 <1> ; 10/01/2026
13502 000105D4 8A86[878D0100] <1> mov al, [esi+p.stat-1]
13503 000105DA 3C00 <1> cmp al, 0
13504 <1> ;cmp byte [esi+p.stat-1], 0 ; SFREE
13505 <1> ; jna short cnpm_2 ; is process active, unused, dead
13506 000105DC 762C <1> jna short cnpm_2 ; it's unused so branch
13507 <1> ;;;
13508 <1> ; 10/01/2026 - TRDOS 386 v2.0.10 (BugFix)
13509 <1> ; (if the parent proc's number = 1, the child's status = 3)
13510 000105DE 3C03 <1> cmp al, 3 ; zombie ?
13511 000105E0 751D <1> jne short cnpm_3 ; al = 1
13512 000105E2 D1E6 <1> shl esi, 1
13513 000105E4 668B86[468D0100] <1> mov ax, [esi+p.ppid-2] ; parent process's id
13514 000105EB D1EE <1> shr esi, 1
13515 000105ED 663B05[288D0100] <1> cmp ax, [p.pid] ; compare with process 1's process id
13516 000105F4 7509 <1> jne short cnpm_3 ; not MainProg (!?)
13517 000105F6 C686[878D0100]00 <1> mov byte [esi+p.stat-1], 0
13518 000105FD EB0B <1> jmp short cnpm_2
13519 <1> cnpm_3:
13520 <1> ;;;
13521 000105FF 6683FE10 <1> cmp si, nproc ; all processes checked
13522 00010603 72CE <1> jb short cnpm_1 ; no, branch back
13523 <1> cnpm_panic:
13524 00010605 E9FF67FFFF <1> jmp panic
13525 <1> cnpm_2:
13526 0001060A A1[AC8E0100] <1> mov eax, [u.pgdir] ; page directory of MainProg
13527 0001060F A3[B08E0100] <1> mov [u.ppgdir], eax ; parent's page directory
13528 00010614 E80D52FFFF <1> call allocate_page
13529 <1> ;jc panic
13530 <1> ; 23/07/2022
13531 00010619 72EA <1> jc short cnpm_panic
13532 <1> ;
13533 <1> ; EAX = UPAGE (user structure page) address
13534 0001061B A3[A88E0100] <1> mov [u.upage], eax ; memory page for 'user' struct (child)
13535 00010620 89F7 <1> mov edi, esi
13536 <1> ;shl di, 2
13537 <1> ; 23/07/2022
13538 00010622 C1E702 <1> shl edi, 2
13539 00010625 8987[948D0100] <1> mov [edi+p.upage-4], eax ; memory page for 'user' struct
13540 0001062B E86752FFFF <1> call clear_page ; 03/05/2016
13541 <1> ;movzx eax, byte [p.ttyc] ; console tty (for MainProg)
13542 <1> ;sub ax, ax ; 0
13543 <1> ; 23/07/2022
13544 00010630 29C0 <1> sub eax, eax
13545 00010632 668986[678D0100] <1> mov [esi+p.ttyc-1], ax ; al - set child's console tty
13546 <1> ; ah - reset child's wait channel
13547 00010639 66A3[888E0100] <1> mov [u.ttyp], ax ; 0
13548 <1> ;
13549 0001063F 89F2 <1> mov edx, esi
13550 00010641 8815[A58E0100] <1> mov [u.uno], dl ; child process number
13551 00010647 FE86[878D0100] <1> inc byte [esi+p.stat-1] ; 1, SRUN
13552 <1> ;shl si, 1 ; multiply si by 2 to get index into p.pid table
13553 <1> ; 23/07/2022
13554 0001064D D1E6 <1> shl esi, 1
13555 0001064F 66FF05[3C8E0100] <1> inc word [mpid] ; increment m.pid; get a new process name
13556 <1> ;
13557 <1> ; 23/07/2022
13558 <1> ;mov ax, [p.pid] ; get process name of MainProg
13559 <1> ;mov ax, 1
13560 00010656 FEC0 <1> inc al ; eax = 1
13561 00010658 668986[468D0100] <1> mov [esi+p.ppid-2], ax ; put parent process name
13562 <1> ; in parent process slot for child
13563 0001065F A2[9E8E0100] <1> mov [u.pri], al ; 1 ; normal priority
13564 <1> ;
13565 <1> ;dec ax ; 0
13566 <1> ;mov [u.ttyp], ax ; 0
13567 <1> ;
13568 00010664 66A1[3C8E0100] <1> mov ax, [mpid]
13569 0001066A 668986[268D0100] <1> mov [esi+p.pid-2], ax ; put new process name
13570 <1> ; in child process' name slot
13571 <1> ;;;
13572 <1> ; 07/07/2025
13573 <1> ;mov eax, [ii]
13574 <1> ; 23/07/2022
13575 <1> ; Retro UNIX 386 v1, 'sysexec' (u2.s)
13576 <1> ;call iopen
13577 <1> ; 06/06/2016
13578 <1> ;mov byte [u.pri], 1 ; normal priority
13579 <1> ;
13580 <1> ;jmp short sysexec_7 ; 02/05/2016
13581 <1> ; 23/07/2022
13582 00010671 E921F9FFFF <1> jmp sysexec_7
13583 <1> ;
13584 <1> ; 02/05/2016
13585 <1> ;inc byte [sysflg] ; 0FFh -> 0
13586 <1> ;mov byte [sysflg], 0 ; 04/01/2017
13587 <1> ;movzx ebx, byte [u.uno]
13588 <1> ;shl bl, 1 ; 13/11/2017
13589 <1> ;cmp word [ebx+p.ppid-2], 1 ; MainProg
13590 <1> ;ja sysret0 ; 03/05/2016
13591 <1> ;push sysret ; *
13592 <1> ;mov [u.usp], esp
13593 <1> ;call wswap ; save child process 'u' structure and
13594 <1> ; registers
13595 <1> ;add dword [u.usp], 4 ; 03/05/2016
13596 <1> ;sysexec_19: ; 02/05/2016
13597 <1> ;retn ; * 'sysret' ; byte [sysflg] -> 0FFh
13598 <1>

```

```

13599      <1> readi:
13600      <1>      ; 09/08/2022
13601      <1>      ; 23/07/2022 - TRDOS 386 Kernel v2.0.5
13602      <1>      ; 01/05/2016
13603      <1>      ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
13604      <1>      ; 20/05/2015 - Retro UNIX 386 v1
13605      <1>      ; 11/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
13606      <1>      ;
13607      <1>      ; Reads from a file whose the first cluster number in EAX
13608      <1>      ;
13609      <1>      ; INPUTS ->
13610      <1>      ; EAX - First cluster number of the file
13611      <1>      ; u.count - byte count user desires
13612      <1>      ; u.base - points to user buffer
13613      <1>      ; u.fofp - points to dword with current file offset
13614      <1>      ; i.size - file size
13615      <1>      ; cdev - logical dos drive number of the file
13616      <1>      ; OUTPUTS ->
13617      <1>      ; u.count - cleared
13618      <1>      ; u.nread - accumulates total bytes passed back
13619      <1>      ;
13620      <1>      ; ((EAX)) input/output
13621      <1>      ; (Retro UNIX Prototype : 14/12/2012 - 01/03/2013, UNIXCOPY.ASM)
13622      <1>      ; ((Modified registers: edx, ebx, ecx, esi, edi))
13623      <1>
13624      00010676 31D2      <1>      xor     edx, edx ; 0
13625      00010678 8915[808E0100] <1>      mov     [u.nread], edx ; 0
13626      0001067E 668915[B88E0100] <1>      mov     [u.pcount], dx ; 19/05/2015
13627      00010685 3915[7C8E0100] <1>      cmp     [u.count], edx ; 0
13628      <1>      ;ja     short readi_1
13629      <1>      ;retn
13630      <1>      ; 09/08/2022
13631      0001068B 765E      <1>      jna     short dskr_5 ; retn
13632      <1>      ;readi_1:
13633      <1>      dskr:
13634      <1>      ; 01/05/2016
13635      <1>      ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
13636      <1>      ; 24/05/2015 - 12/10/2015 (Retro UNIX 386 v1)
13637      <1>      ; 26/04/2013 - 03/08/2013 (Retro UNIX 8086 v1)
13638      <1>      dskr_0:
13639      0001068D 8B15[608F0100] <1>      mov     edx, [i.size]
13640      00010693 8B1D[688E0100] <1>      mov     ebx, [u.fofp]
13641      00010699 2B13      <1>      sub     edx, [ebx]
13642      0001069B 7647      <1>      jna     short dskr_4
13643      <1>      ;
13644      0001069D 50        <1>      push    eax ; 01/05/2016
13645      0001069E 3B15[7C8E0100] <1>      cmp     edx, [u.count]
13646      000106A4 7306      <1>      jnb     short dskr_1
13647      000106A6 8915[7C8E0100] <1>      mov     [u.count], edx
13648      <1>      dskr_1:
13649      <1>      ; EAX = First Cluster
13650      <1>      ; [Current_Drv] = Physical drive number
13651      000106AC E83B000000 <1>      call    mget_r
13652      <1>      ; NOTE: in 'mget_r', relevant sector will be read in buffer
13653      <1>      ; if it is not already in buffer !
13654      000106B1 BB[7C900100] <1>      mov     ebx, readi_buffer
13655      000106B6 803D[BA8E0100]00 <1>      cmp     byte [u.kcall], 0 ; the caller is 'namei' sign (=1)
13656      000106BD 770F      <1>      ja      short dskr_3 ; zf=0 -> the caller is 'namei'
13657      000106BF 66833D[B88E0100]00 <1>      cmp     word [u.pcount], 0
13658      000106C7 7705      <1>      ja      short dskr_3
13659      <1>      dskr_2:
13660      <1>      ; [u.base] = virtual address to transfer (as destination address)
13661      000106C9 E869010000 <1>      call    trans_addr_w ; translate virtual address to physical (w)
13662      <1>      dskr_3:
13663      <1>      ; EBX (r5) = system (I/O) buffer address -physical-
13664      000106CE E8CC010000 <1>      call    storeg
13665      000106D3 87FE      <1>      xchg     esi, edi
13666      <1>      ; EDI = file (user data) offset
13667      <1>      ; ESI = sector (I/O) buffer offset
13668      <1>      ; ECX = byte count
13669      000106D5 F3A4      <1>      rep     movsb
13670      <1>      ; eax = remain bytes in buffer
13671      <1>      ; (check if remain bytes in the buffer > [u.pcount])
13672      000106D7 09C0      <1>      or      eax, eax
13673      000106D9 75EE      <1>      jnz     short dskr_2 ; (page end before system buffer end!)
13674      000106DB 58        <1>      pop     eax ; (first cluster number)
13675      000106DC 390D[7C8E0100] <1>      cmp     [u.count], ecx ; 0
13676      000106E2 77A9      <1>      ja      short dskr_0
13677      <1>      dskr_4:
13678      000106E4 C605[BA8E0100]00 <1>      mov     byte [u.kcall], 0
13679      <1>      dskr_5:
13680      000106EB C3        <1>      retn
13681      <1>
13682      <1> mget_r:
13683      <1>      ; 19/12/2025 - TRDOS 386 kernel v2.0.10
13684      <1>      ; 29/08/2023
13685      <1>      ; 30/07/2022
13686      <1>      ; 23/07/2022 - TRDOS 386 Kernel v2.0.5
13687      <1>      ; 24/10/2016
13688      <1>      ; 22/10/2016
13689      <1>      ; 12/10/2016
13690      <1>      ; 29/04/2016
13691      <1>      ; 25/04/2016 - TRDOS 386 (TRDOS v2.0)
13692      <1>      ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)
13693      <1>      ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
13694      <1>      ;
13695      <1>      ; Get existing or (allocate) a new disk block for file
13696      <1>      ;
13697      <1>      ; INPUTS ->
13698      <1>      ; [u.fofp] = file offset pointer
13699      <1>      ; EAX = First Cluster
13700      <1>      ; [cdev] = Logical dos drive number
13701      <1>      ; ([u.off] = file offset)
13702      <1>      ; OUTPUTS ->
13703      <1>      ; EAX = logical sector number
13704      <1>      ; ESI = Logical Dos Drive Description Table address
13705      <1>      ;
13706      <1>      ; Modified registers: EDX, EBX, ECX, ESI, EDI
13707      <1>
13708      000106EC 8B35[688E0100] <1>      mov     esi, [u.fofp]
13709      000106F2 8B1E      <1>      mov     ebx, [esi] ; (u.off)
13710      <1>
13711      000106F4 29C9      <1>      sub     ecx, ecx
13712      000106F6 8A2D[398E0100] <1>      mov     ch, [cdev]
13713      <1>
13714      000106FC BE00010900 <1>      mov     esi, Logical_DOSDisks
13715      00010701 01CE      <1>      add     esi, ecx
13716      <1>
13717      00010703 380D[D0820100] <1>      cmp     [readi.valid], cl ; 0
13718      00010709 7642      <1>      jna     short mget_r_0
13719      <1>
13720      0001070B 3A2D[D1820100] <1>      cmp     ch, [readi.driv]
13721      00010711 753A      <1>      jne     short mget_r_0
13722      <1>

```

```

13723 00010713 3B05[E4820100] <1> cmp     eax, [readi.fclust]
13724 00010719 754E <1> jne     short mget_r_3
13725 <1>
13726 0001071B 89D8 <1> mov     eax, ebx ; file offset
13727 0001071D 668B0D[D8820100] <1> mov     cx, [readi.bpc]
13728 00010724 41 <1> inc     ecx ; <= 65536
13729 00010725 29D2 <1> sub     edx, edx
13730 00010727 F7F1 <1> div     ecx
13731 <1>
13732 00010729 8B3D[E0820100] <1> mov     edi, [readi.c_index] ; cluster index
13733 <1>
13734 0001072F 39F8 <1> cmp     eax, edi
13735 00010731 7563 <1> jne     short mget_r_4 ; (*)
13736 <1>
13737 <1> ; edx = byte offset in cluster (<= 65535)
13738 00010733 668915[DA820100] <1> mov     [readi.offset], dx
13739 <1> ; 23/07/2022
13740 <1> ;shr     dx, 9 ; / 512
13741 0001073A C1EA09 <1> shr     edx, 9
13742 0001073D 8815[D3820100] <1> mov     [readi.s_index], dl ; sector index in cluster (0 to spc -1)
13743 <1>
13744 00010743 A1[DC820100] <1> mov     eax, [readi.cluster] ; > 0 if [readi.valid] = 1
13745 <1> ; 19/12/2025
13746 <1> ;mov     edx, [readi.fs_index]
13747 00010748 E989000000 <1> jmp     mget_r_7
13748 <1>
13749 <1> mget_r_0:
13750 0001074D 882D[D1820100] <1> mov     [readi.driv], ch ; physical drive number
13751 <1> ; 19/12/2025
13752 <1> %if 0
13753 <1> cmp     byte [esi+LD_FATType], 0
13754 <1> ja      short mget_r_1
13755 <1> ;mov     cl, [esi+LD_FS_BytesPerSec+1]
13756 <1> ;shr     cl, 1 ; 1 for 512 bytes, 4 for 2048 bytes
13757 <1> ; 19/12/2025
13758 <1> mov     cl, 1
13759 <1> jmp     short mget_r_2
13760 <1> %endif
13761 <1> mget_r_1:
13762 00010753 8A4E13 <1> mov     cl, [esi+LD_BPB+BPB_SecPerClust]
13763 <1> mget_r_2:
13764 00010756 880D[D2820100] <1> mov     [readi.spc], cl ; sectors per cluster
13765 <1> ; NOTE: readi bytes per sector value is always 512 !
13766 <1> ; 23/07/2022
13767 <1> ;xor     ch, ch
13768 <1> ; 29/08/2023
13769 0001075C C1E109 <1> shl     ecx, 9
13770 <1> ;shr     cx, 9 ; * 512
13771 <1> ;dec     cx ; bytes per cluster - 1
13772 <1> ; 23/07/2022
13773 0001075F 49 <1> dec     ecx
13774 00010760 66890D[D8820100] <1> mov     [readi.bpc], cx
13775 <1> ;sub     cx, cx
13776 <1> ; 23/07/2022
13777 00010767 29C9 <1> sub     ecx, ecx
13778 <1> mget_r_3:
13779 00010769 A3[E4820100] <1> mov     [readi.fclust], eax ; first cluster (or FDT address)
13780 0001076E 880D[D0820100] <1> mov     [readi.valid], cl ; 0
13781 <1> ;mov     [readi.s_index], cl ; 0
13782 <1> ;mov     [readi.offset], cx ; 0
13783 00010774 890D[E0820100] <1> mov     [readi.c_index], ecx ; 0
13784 0001077A 890D[DC820100] <1> mov     [readi.cluster], ecx ; 0
13785 00010780 890D[D4820100] <1> mov     [readi.sector], ecx ; 0
13786 <1>
13787 00010786 89D8 <1> mov     eax, ebx ; file offset
13788 00010788 668B0D[D8820100] <1> mov     cx, [readi.bpc]
13789 0001078F 41 <1> inc     ecx ; <= 65536
13790 00010790 29D2 <1> sub     edx, edx
13791 00010792 F7F1 <1> div     ecx
13792 <1> ;mov     edi, [readi.c_index] ; previous cluster index
13793 00010794 29FF <1> sub     edi, edi
13794 <1> mget_r_4:
13795 00010796 A3[E0820100] <1> mov     [readi.c_index], eax ; cluster index
13796 <1> ; edx = byte offset in cluster (<= 65535)
13797 0001079B 668915[DA820100] <1> mov     [readi.offset], dx
13798 <1> ; 23/07/2022
13799 <1> ;shr     dx, 9 ; / 512
13800 000107A2 C1EA09 <1> shr     edx, 9
13801 000107A5 8815[D3820100] <1> mov     [readi.s_index], dl ; sector index in cluster (0 to spc -1)
13802 <1>
13803 000107AB 89C1 <1> mov     ecx, eax ; current cluster index
13804 000107AD A1[E4820100] <1> mov     eax, [readi.fclust]
13805 000107B2 09C9 <1> or      ecx, ecx ; cluster index
13806 000107B4 741B <1> jz      short mget_r_6
13807 <1>
13808 000107B6 39CF <1> cmp     edi, ecx
13809 000107B8 7710 <1> ja      short mget_r_5 ; old cluster index is higher
13810 000107BA 8B15[DC820100] <1> mov     edx, [readi.cluster]
13811 000107C0 21D2 <1> and     edx, edx
13812 000107C2 7406 <1> jz      short mget_r_5
13813 <1> ; valid 'readi' parameters (*)
13814 000107C4 89D0 <1> mov     eax, edx
13815 000107C6 29F9 <1> sub     ecx, edi
13816 000107C8 740C <1> jz      short mget_r_7
13817 <1> mget_r_5:
13818 <1> ; EAX = Beginning cluster
13819 <1> ; EDX = Sector index in disk/file section
13820 <1> ; (Only for SINGLIX file system!)
13821 <1> ; ECX = Cluster sequence number after the beginning cluster
13822 <1> ; ESI = Logical DOS Drive Description Table address
13823 000107CA E8DAC0FFFF <1> call    get_cluster_by_index
13824 000107CF 7246 <1> jc      short mget_r_err
13825 <1> ; EAX = Cluster number
13826 <1> mget_r_6:
13827 000107D1 A3[DC820100] <1> mov     [readi.cluster], eax ; FDT number for Singlix File System
13828 <1> mget_r_7:
13829 <1> ; 19/12/2025
13830 <1> %if 0
13831 <1> cmp     byte [esi+LD_FATType], 0
13832 <1> jna     short mget_r_12
13833 <1> %endif
13834 <1> ;sub     eax, 2
13835 <1> ; 30/07/2022
13836 000107D6 48 <1> dec     eax
13837 000107D7 48 <1> dec     eax
13838 000107D8 0FB615[D2820100] <1> movzx   edx, byte [readi.spc]
13839 000107DF F7E2 <1> mul     edx
13840 <1>
13841 000107E1 034668 <1> add     eax, [esi+LD_DATABegin]
13842 000107E4 8A15[D3820100] <1> mov     dl, [readi.s_index]
13843 000107EA 01D0 <1> add     eax, edx
13844 <1> mget_r_8:
13845 <1> ; eax = logical sector number
13846 000107EC 803D[D0820100]00 <1> cmp     byte [readi.valid], 0

```

```

13847 000107F3 7608      <1>      jna      short mget_r_9
13848 000107F5 3805[D4820100] <1>      cmp      eax, [readi.sector]
13849 000107FB 7435      <1>      je       short mget_r_11 ; sector is already in 'readi' buffer
13850                                <1> mget_r_9:
13851 000107FD A3[D4820100] <1>      mov      [readi.sector], eax
13852 00010802 BB[7C900100] <1>      mov      ebx, readi_buffer ; buffer address
13853                                <1>      ;mov      ecx, 1
13854                                <1>      ; 30/07/2022
13855 00010807 31C9      <1>      xor      ecx, ecx
13856 00010809 FEC1      <1>      inc      cl
13857                                <1>      ; ecx = 1
13858                                <1>
13859                                <1>      ; 29/04/2016
13860                                <1>      ;xor      dl, dl
13861                                <1>
13862                                <1>      ; EAX = Logical sector number
13863                                <1>      ; ECX = Sector count
13864                                <1>      ; EBX = Buffer address
13865                                <1>      ; (EDX = 0)
13866                                <1>      ; ESI = Logical DOS drive description table address
13867                                <1>
13868 0001080B E8D3110000 <1>      call     disk_read
13869 00010810 7314      <1>      jnc      short mget_r_10
13870                                <1>
13871                                <1>      ; 22/10/2016 (15h -> 17)
13872 00010812 B811000000 <1>      mov      eax, 17 ; Drive not ready or read error !
13873                                <1> mget_r_err:
13874 00010817 A3[C08E0100] <1>      mov      [u.error], eax
13875                                <1>      ; 12/10/2016
13876 0001081C A3[548E0100] <1>      mov      [u.r0], eax
13877 00010821 E938C2FFFF <1>      jmp      error
13878                                <1> mget_r_10:
13879 00010826 C605[D0820100]01 <1>      mov      byte [readi.valid], 1 ; 24/10/2016
13880 0001082D A1[D4820100] <1>      mov      eax, [readi.sector]
13881                                <1> mget_r_11:
13882 00010832 C3          <1>      retn
13883                                <1>
13884                                <1>      ; 19/12/2025
13885                                <1> %if 0
13886                                <1> mget_r_12:
13887                                <1>      ; EAX = FDT number
13888                                <1>      ; EDX = Sector index from FDT sector (0,1,2,3,4...)
13889                                <1>      inc      eax ; the first data sector in FS disk section
13890                                <1>      mov      [readi.fs_index], edx
13891                                <1>      add      eax, edx
13892                                <1>      jmp      short mget_r_8
13893                                <1> %endif
13894                                <1>
13895                                <1> trans_addr_r:
13896                                <1>      ; 12/10/2016
13897                                <1>      ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)
13898                                <1>      ; Translate virtual address to physical address
13899                                <1>      ; for reading from user's memory space
13900                                <1>      ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)
13901                                <1>
13902 00010833 31D2      <1>      xor      edx, edx ; 0 (read access sign)
13903 00010835 EB04      <1>      jmp      short trans_addr_rw
13904                                <1>
13905                                <1> trans_addr_w:
13906                                <1>      ; 12/10/2016
13907                                <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
13908                                <1>      ; Translate virtual address to physical address
13909                                <1>      ; for writing to user's memory space
13910                                <1>      ; 04/06/2015 - 18/10/2015 (Retro UNIX 386 v1)
13911                                <1>
13912 00010837 29D2      <1>      sub      edx, edx
13913 00010839 FEC2      <1>      inc      dl ; 1 (write access sign)
13914                                <1> trans_addr_rw:
13915 0001083B 50          <1>      push     eax
13916 0001083C 53          <1>      push     ebx
13917 0001083D 52          <1>      push     edx ; r/w sign (in DL)
13918                                <1>      ;
13919 0001083E 8B1D[788E0100] <1>      mov      ebx, [u.base]
13920 00010844 E8B53FFFFF <1>      call     get_physical_addr ; get physical address
13921 00010849 730F      <1>      jnc      short passc_0
13922 0001084B A3[C08E0100] <1>      mov      [u.error], eax
13923 00010850 A3[548E0100] <1>      mov      [u.r0], eax ; 12/10/2016
13924                                <1>      ;pop      edx
13925                                <1>      ;pop      ebx
13926                                <1>      ;pop      eax
13927 00010855 E904C2FFFF <1>      jmp      error
13928                                <1> passc_0:
13929 0001085A F6C202      <1>      test     dl, PTE_A_WRITE ; writable page
13930 0001085D 5A          <1>      pop      edx
13931 0001085E 751C      <1>      jnz      short passc_1
13932                                <1>
13933 00010860 20D2      <1>      and      dl, dl
13934 00010862 7418      <1>      jz       short passc_1
13935                                <1>      ; read only (duplicated) page -must be copied to a new page-
13936                                <1>      ; EBX = linear address
13937 00010864 51          <1>      push     ecx
13938 00010865 E80B53FFFF <1>      call     copy_page
13939 0001086A 59          <1>      pop      ecx
13940 0001086B 721E      <1>      jc       short passc_2
13941 0001086D 50          <1>      push     eax ; physical address of the new/allocated page
13942 0001086E E89153FFFF <1>      call     add_to_swap_queue
13943 00010873 58          <1>      pop      eax
13944 00010874 81E3FF0F0000 <1>      and      ebx, PAGE_OFF ; 0FFFFh
13945                                <1>      ;mov      ecx, PAGE_SIZE
13946                                <1>      ;sub      ecx, ebx
13947 0001087A 01D8      <1>      add      eax, ebx
13948                                <1> passc_1:
13949 0001087C A3[B48E0100] <1>      mov      [u.pbase], eax ; physical address
13950 00010881 66890D[B88E0100] <1>      mov      [u.pcount], cx ; remain byte count in page (1-4096)
13951 00010888 5B          <1>      pop      ebx
13952 00010889 58          <1>      pop      eax
13953 0001088A C3          <1>      retn
13954                                <1> passc_2:
13955 0001088B B804000000 <1>      mov      eax, ERR_MINOR_IM ; "Insufficient memory !" error
13956 00010890 A3[548E0100] <1>      mov      [u.r0], eax ; 12/10/2016
13957 00010895 A3[C08E0100] <1>      mov      [u.error], eax
13958                                <1>      ;pop      ebx
13959                                <1>      ;pop      eax
13960 0001089A E9BFC1FFFF <1>      jmp      error
13961                                <1>
13962                                <1> sioreg:
13963                                <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
13964                                <1>      ; 19/05/2015 - 25/07/2015 (Retro UNIX 386 v1)
13965                                <1>      ; 12/03/2013 - 22/07/2013 (Retro UNIX 8086 v1)
13966                                <1>      ; INPUTS ->
13967                                <1>      ; EBX = system buffer (data) address (r5)
13968                                <1>      ; [u.fotp] = pointer to file offset pointer
13969                                <1>      ; [u.base] = virtual address of the user buffer
13970                                <1>      ; [u.pbase] = physical address of the user buffer

```



```

13971      <1>      ;      [u.count] = byte count
13972      <1>      ;      [u.pcount] = byte count within page frame
13973      <1>      ;      OUTPUTS ->
13974      <1>      ;      ESI = user data offset (r1)
13975      <1>      ;      EDI = system (I/O) buffer offset (r2)
13976      <1>      ;      ECX = byte count (r3)
13977      <1>      ;      EAX = remain bytes after byte count within page frame
13978      <1>      ;      (If EAX > 0, transfer will continue from the next page)
13979      <1>      ;
13980      <1>      ; ((Modified registers: EDX))
13981      <1>
13982      0001089F 8B35[688E0100]      <1>      mov     esi, [u.fofp]
13983      000108A5 8B3E      <1>      mov     edi, [esi]
13984      000108A7 89F9      <1>      mov     ecx, edi
13985      000108A9 81C900FEFFFF      <1>      or      ecx, 0FFFFFF0h
13986      000108AF 81E7FF010000      <1>      and     edi, 1FFh
13987      000108B5 01DF      <1>      add     edi, ebx ; EBX = system buffer (data) address
13988      000108B7 F7D9      <1>      neg     ecx
13989      000108B9 3B0D[7C8E0100]      <1>      cmp     ecx, [u.count]
13990      000108BF 7606      <1>      jna     short sioreg_0
13991      000108C1 8B0D[7C8E0100]      <1>      mov     ecx, [u.count]
13992      <1>      sioreg_0:
13993      000108C7 803D[BA8E0100]00      <1>      cmp     byte [u.kcall], 0
13994      000108CE 7613      <1>      jna     short sioreg_1
13995      <1>      ; the caller is 'mkdir' or 'namei'
13996      000108D0 A1[788E0100]      <1>      mov     eax, [u.base]
13997      000108D5 A3[B48E0100]      <1>      mov     [u.pbase], eax ; physical address = virtual address
13998      000108DA 66890D[B88E0100]      <1>      mov     word [u.pcount], cx ; remain bytes in buffer (1 sector)
13999      000108E1 EB0B      <1>      jmp     short sioreg_2
14000      <1>      sioreg_1:
14001      000108E3 0FB715[B88E0100]      <1>      movzx   edx, word [u.pcount]
14002      000108EA 39D1      <1>      cmp     ecx, edx
14003      000108EC 772A      <1>      ja      short sioreg_4 ; transfer count > [u.pcount]
14004      <1>      sioreg_2: ; 2:
14005      000108EE 31C0      <1>      xor     eax, eax
14006      <1>      sioreg_3:
14007      000108F0 010D[808E0100]      <1>      add     [u.nread], ecx
14008      000108F6 290D[7C8E0100]      <1>      sub     [u.count], ecx
14009      000108FC 010D[788E0100]      <1>      add     [u.base], ecx
14010      00010902 010E      <1>      add     [esi], ecx
14011      00010904 8B35[B48E0100]      <1>      mov     esi, [u.pbase]
14012      0001090A 66290D[B88E0100]      <1>      sub     [u.pcount], cx
14013      00010911 010D[B48E0100]      <1>      add     [u.pbase], ecx
14014      00010917 C3      <1>      retn
14015      <1>      sioreg_4:
14016      <1>      ; transfer count > [u.pcount]
14017      <1>      ; (ecx > edx)
14018      00010918 89C8      <1>      mov     eax, ecx
14019      0001091A 29D0      <1>      sub     eax, edx ; remain bytes for 1 sector (block) transfer
14020      0001091C 89D1      <1>      mov     ecx, edx ; current transfer count = [u.pcount]
14021      0001091E EBD0      <1>      jmp     short sioreg_3
14022      <1>
14023      <1>      tswitch: ; Retro UNIX 386 v1
14024      <1>      tswap:
14025      <1>      ; 16/01/2017
14026      <1>      ; 21/05/2016 - TRDOS 386 (TRDOS v2.0)
14027      <1>      ; 10/05/2015 - 01/09/2015 (Retro UNIX 386 v1)
14028      <1>      ; 14/04/2013 - 14/02/2014 (Retro UNIX 8086 v1)
14029      <1>      ; time out swap, called when a user times out.
14030      <1>      ; the user is put on the low priority queue.
14031      <1>      ; This is done by making a link from the last user
14032      <1>      ; on the low priority queue to him via a call to 'putlu'.
14033      <1>      ; then he is swapped out.
14034      <1>
14035      <1>      ; TRDOS 386 (TRDOS v2.0) modification -> ** 21/05/2016 **
14036      <1>      ; * when a high priority (event) process will be stopped
14037      <1>      ; (swapped out, switched out/off), 'tswap/tswitch' will
14038      <1>      ; not add it to a run queue.
14039      <1>      ; /// what for: Process may be already in a run queue,
14040      <1>      ; it is unspecificed state because process might be started
14041      <1>      ; by a timer event which does not regard previous priority
14042      <1>      ; level and run queue of the process (for fast executing!).
14043      <1>      ; After the 'run for event', process will be sequenced
14044      <1>      ; to run by it's actual run queue. ///
14045      <1>
14046      <1>      ; Retro UNIX 386 v1 modification ->
14047      <1>      ; swap (software task switch) is performed by changing
14048      <1>      ; user's page directory (u.pgdir) instead of segment change
14049      <1>      ; as in Retro UNIX 8086 v1.
14050      <1>
14051      <1>      ; RETRO UNIX 8086 v1 modification ->
14052      <1>      ; 'swap to disk' is replaced with 'change running segment'
14053      <1>      ; according to 8086 cpu (x86 real mode) architecture.
14054      <1>      ; pdp-11 was using 64KB uniform memory while IBM PC
14055      <1>      ; compatibles was using 1MB segmented memory
14056      <1>      ; in 8086/8088 times.
14057      <1>
14058      <1>      ; INPUTS ->
14059      <1>      ; u.uno - users process number
14060      <1>      ; runq+4 - lowest priority queue
14061      <1>      ; OUTPUTS ->
14062      <1>      ; r0 - users process number
14063      <1>      ; r2 - lowest priority queue address
14064      <1>
14065      <1>      ; ((AX = R0, BX = R2)) output
14066      <1>      ; ((Modified registers: EDX, EBX, ECX, ESI, EDI))
14067      <1>
14068      <1>
14069      <1>      NOTE:
14070      <1>      ; * [u.pri] priority level is specified by run queue which is process
14071      <1>      ; comes to run from.
14072      <1>      ; * Initial [u.pri] is 1 ('normal/regular') for programs
14073      <1>      ; (which are launched by MainProg or 'sysexec'), it is changed
14074      <1>      ; to 2 ('high') by timer event, if program uses 'systimer' system call.
14075      <1>      ; * Program (Process) also can change it's running priority
14076      <1>      ; from 1 to 0 or up to 2 by using 'syspri' system call; but,
14077      <1>      ; if program selects priority level 2 (high) for running, next time
14078      <1>      ; it is reduced to 1 (normal/regular) because 'syspri' adds this
14079      <1>      ; program to 'run for normal' queue while running duration is a bit
14080      <1>      ; protected from swap/switch out immediate, behalf of other high
14081      <1>      ; priority process in sequence. Program (with high priority) will not
14082      <1>      ; be swapped/switched out (by timer event) before it's time quantum
14083      <1>      ; will be elapsed, but, this will be temporary if program is not using
14084      <1>      ; timer event function.
14085      <1>
14086      <1>      ; For example:
14087      <1>      ; If a process frequently gets a timer event, it runs at high priority
14088      <1>      ; level but when it returns from running it returns to actual run queue,
14089      <1>      ; not to 'run for event' queue again.
14090      <1>      ; 'tswap' will not change the sequence at return/stop(swap out) stage.
14091      <1>      ; But if priority level not high (=2, 'run for event'), 'tswap/tswitch'
14092      <1>      ; will add the stopping process to relevant run queue according to
14093      <1>      ; [u.pri] priority level.
14094      <1>

```

```

14095      <1>      ; 16/01/2017
14096 00010920 BB[448E0100] <1>      mov     ebx, runq+2      ; 'runq_normal' ; normal/regular priority
14097      <1>      ; 21/05/2016
14098      <1>      ; cmp     byte [u.pri], 2      ; high priority (run for event) ?
14099      <1>      ; jnb     short swap
14100      <1>      ; 16/01/2017
14101      <1>      ; (Normal and also high/event priority processes will be added to
14102      <1>      ; normal priority run queue for ensuring circular running sequence!)
14103      <1>      ; (Timer interrupt or 'syspri' system call may change priority and run
14104      <1>      ; queue to high/event level.)
14105 00010925 803D[9E8E0100]00 <1>      cmp     byte [u.pri], 0
14106 0001092C 7702 <1>      ja      short tswap_1      ; normal priority run queue
14107      <1>      ;
14108 0001092E 43 <1>      inc     ebx
14109 0001092F 43 <1>      inc     ebx      ; runq+4, 'runq_background', low priority
14110      <1>      tswap_1:
14111 00010930 A0[A58E0100] <1>      mov     al, [u.uno]
14112      <1>      ; movb u.uno,r1 / move users process number to r1
14113      <1>      ; mov     $runq+4,r2
14114      <1>      ; / move lowest priority queue address to r2
14115      <1>      ; ebx = run queue
14116 00010935 E8FD000000 <1>      call    putlu
14117      <1>      ; jsr r0,putlu / create link from last user on Q to
14118      <1>      ; / u.uno's user
14119      <1>
14120      <1>      switch: ; Retro UNIX 386 v1
14121      <1>      swap:
14122      <1>      ; 20/08/2024
14123      <1>      ; 02/01/2017
14124      <1>      ; 21/05/2016
14125      <1>      ; 20/05/2016
14126      <1>      ; 02/05/2016
14127      <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
14128      <1>      ; 10/05/2015 - 02/09/2015 (Retro UNIX 386 v1)
14129      <1>      ; 14/04/2013 - 08/03/2014 (Retro UNIX 8086 v1)
14130      <1>
14131      <1>      ; 'swap' is routine that controls the swapping of processes
14132      <1>      ; in and out of core.
14133      <1>
14134      <1>      ; TRDOS 386 (TRDOS v2.0) modification -> ** 20/05/2016 **
14135      <1>      ; * 3 different priority level is applied
14136      <1>      ; (just as original unix v1)
14137      <1>      ; 1) high priority (event) run queue, 'runq_event'
14138      <1>      ; 2) normal priority (regular) run queue, 'runq_normal'
14139      <1>      ; 3) low priority (background) run queue, 'runq_background'
14140      <1>      ; 'swap' code will run a process which has max. priority
14141      <1>      ; (for earliest event at first)
14142      <1>
14143      <1>      ; Retro UNIX 386 v1 modification ->
14144      <1>      ; swap (software task switch) is performed by changing
14145      <1>      ; user's page directory (u.pgdir) instead of segment change
14146      <1>      ; as in Retro UNIX 8086 v1.
14147      <1>
14148      <1>      ; RETRO UNIX 8086 v1 modification ->
14149      <1>      ; 'swap to disk' is replaced with 'change running segment'
14150      <1>      ; according to 8086 cpu (x86 real mode) architecture.
14151      <1>      ; pdp-11 was using 64KB uniform memory while IBM PC
14152      <1>      ; compatibles was using 1MB segmented memory
14153      <1>      ; in 8086/8088 times.
14154      <1>
14155      <1>      ; INPUTS ->
14156      <1>      ; runq table - contains processes to run.
14157      <1>      ; p.link - contains next process in line to be run.
14158      <1>      ; u.uno - process number of process in core
14159      <1>      ; s.stack - swap stack used as an internal stack for swapping.
14160      <1>      ; OUTPUTS ->
14161      <1>      ; (original unix v1 -> present process to its disk block)
14162      <1>      ; (original unix v1 -> new process into core ->
14163      <1>      ; Retro unix 8086 v1 -> segment registers changed
14164      <1>      ; for new process)
14165      <1>      ; u.quant = 3 (Time quantum for a process)
14166      <1>      ; ((INT 1Ch count down speed -> 18.2 times per second)
14167      <1>      ; RETRO UNIX 8086 v1 will use INT 1Ch (18.2 times per second)
14168      <1>      ; for now, it will swap the process if there is not
14169      <1>      ; a keyboard event (keystroke) (Int 15h, function 4Fh)
14170      <1>      ; or will count down from 3 to 0 even if there is a
14171      <1>      ; keyboard event locking due to repetitive key strokes.
14172      <1>      ; u.quant will be reset to 3 for RETRO UNIX 8086 v1.
14173      <1>
14174      <1>      ; ((Modified registers: EAX, EDX, EBX, ECX, ESI, EDI))
14175      <1>
14176      <1>      ;NOTE:
14177      <1>      ;High priority queue is the first for selecting a process to run.
14178      <1>      ;If there is not a process in high priority level run queue,
14179      <1>      ;a process in normal priority run queue will be selected
14180      <1>      ;or a proces in low priority run queue will be selected if normal
14181      <1>      ;priority level run queue is empty.
14182      <1>
14183      <1>      ; 21/05/2016 -(3 priority levels, 3 run queues)
14184 0001093A BE[428E0100] <1>      mov     esi, runq ; 'runq_event' ; high priority, 'run for event'
14185 0001093F C605[30830100]03 <1>      mov     byte [priority], 3 ; high priority + 1
14186 00010946 31DB <1>      xor     ebx, ebx ; 02/01/2017
14187      <1>      swap_0: ; 1: / search runq table for highest priority process
14188 00010948 66AD <1>      lodsw    ; mov ax, [esi], add esi+2
14189      <1>      ;xor     ebx, ebx ; 02/05/2016
14190 0001094A 6621C0 <1>      and     ax, ax ; are there any processes to run in this Q entry
14191 0001094D 750E <1>      jnz     short swap_2
14192      <1>      ; 21/05/2026
14193      <1>      ; runq_normal = runq+2, runq_background = runq+4
14194 0001094F FE0D[30830100] <1>      dec     byte [priority] ; 3 -> 3, 2 -> 1, 1 -> 0
14195 00010955 75F1 <1>      jnz     short swap_0
14196      <1>      ; cmp     esi, runq+6 ; if zero compare address to end of table
14197      <1>      ; jnb     short swap_0 ; if not at end, go back
14198      <1>      swap_1:
14199      <1>      ; 02/05/2016
14200      <1>      ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
14201      <1>      ; No user process to run...
14202      <1>      ; Run the kernel process... MainProg: Internal Command Interpreter
14203 00010957 FEC0 <1>      inc     al ; mov al, 1 ; process number of MainProg
14204 00010959 FEC3 <1>      inc     bl ; mov bl, al ; 1
14205 0001095B EB1D <1>      jmp     short swap_4
14206      <1>      swap_2:
14207      <1>      ; 21/05/2016
14208 0001095D FE0D[30830100] <1>      dec     byte [priority] ; priority level of present user/process
14209      <1>      ; 0, 1, 2
14210 00010963 4E <1>      dec     esi
14211 00010964 4E <1>      dec     esi
14212      <1>      ;
14213 00010965 88C3 <1>      mov     bl, al
14214 00010967 38E0 <1>      cmp     al, ah ; is there only 1 process in the queue to be run
14215 00010969 740A <1>      je      short swap_3 ; yes
14216 0001096B 8AA3[778D0100] <1>      mov     ah, [ebx+p.link-1]
14217 00010971 8826 <1>      mov     [esi], ah ; move next process in line into run queue
14218 00010973 EB05 <1>      jmp     short swap_4

```

```

14219 <1> swap_3:
14220 <1> ;xor dx, dx
14221 <1> ; 20/08/2024
14222 00010975 31D2 <1> xor edx, edx
14223 00010977 668916 <1> mov [esi], dx ; zero the entry; no processes on the Q
14224 <1> swap_4:
14225 0001097A 8A25[A58E0100] <1> mov ah, [u.uno]
14226 00010980 38C4 <1> cmp ah, al ; is this process the same as the process in core?
14227 00010982 743B <1> je short swap_8 ; yes, don't have to swap
14228 00010984 08E4 <1> or ah, ah ; is the process # = 0
14229 00010986 740D <1> jz short swap_6 ; 'sysexit'
14230 <1> ;cmp ah, al ; is this process the same as the process in core?
14231 <1> ;je short swap_8 ; yes, don't have to swap
14232 00010988 8925[508E0100] <1> mov [u.usp], esp ; return address for 'syswait' & 'sleep'
14233 0001098E E834000000 <1> call wswap ; write out core to disk
14234 00010993 EB1C <1> jmp short swap_7
14235 <1> swap_6:
14236 <1> ; Deallocate memory pages belong to the process
14237 <1> ; which is being terminated.
14238 <1> ; (Retro UNIX 386 v1 modification !)
14239 <1> ;
14240 00010995 53 <1> push ebx
14241 00010996 A1[AC8E0100] <1> mov eax, [u.pgdir] ; page directory of the process
14242 0001099B 8B1D[B08E0100] <1> mov ebx, [u.pgdir] ; page directory of the parent process
14243 000109A1 E8B04FFFFF <1> call deallocate_page_dir
14244 000109A6 A1[A88E0100] <1> mov eax, [u.upage] ; 'user' structure page of the process
14245 000109AB E83550FFFF <1> call deallocate_page
14246 000109B0 5B <1> pop ebx
14247 <1> swap_7:
14248 000109B1 C0E302 <1> shl bl, 2 ; * 4
14249 <1> ;;;
14250 000109B4 8B83[948D0100] <1> mov eax, [ebx+p.upage-4] ; the 'u' page of the new process
14251 000109BA E840000000 <1> call rswap ; read new process into core
14252 <1> swap_8:
14253 <1> ; Retro UNIX 8086 v1 modification !
14254 000109BF C605[9C8E0100]04 <1> mov byte [u.quant], time_count
14255 000109C6 C3 <1> retn
14256 <1>
14257 <1> wswap: ; < swap out, swap to disk >
14258 <1> ; 28/02/2017 (fnsave)
14259 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
14260 <1> ; 09/05/2015 (Retro UNIX 386 v1)
14261 <1> ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
14262 <1> ; 'wswap' writes out the process that is in core onto its
14263 <1> ; appropriate disk area.
14264 <1> ;
14265 <1> ; Retro UNIX 386 v1 modification ->
14266 <1> ; User (u) structure content and the user's register content
14267 <1> ; will be copied to the process's/user's UPAGE (a page for
14268 <1> ; saving 'u' structure and user registers for task switching).
14269 <1> ; u.usp - points to kernel stack address which contains
14270 <1> ; user's registers while entering system call.
14271 <1> ; u.sp - points to kernel stack address
14272 <1> ; to return from system call -for IRET-.
14273 <1> ; [u.usp]+32+16 = [u.sp]
14274 <1> ; [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
14275 <1> ; edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
14276 <1> ;
14277 <1> ; Retro UNIX 8086 v1 modification ->
14278 <1> ; 'swap to disk' is replaced with 'change running segment'
14279 <1> ; according to 8086 cpu (x86 real mode) architecture.
14280 <1> ; pdp-11 was using 64KB uniform memory while IBM PC
14281 <1> ; compatibles was using 1MB segmented memory
14282 <1> ; in 8086/8088 times.
14283 <1> ;
14284 <1> ; INPUTS ->
14285 <1> ; u.break - points to end of program
14286 <1> ; u.usp - stack pointer at the moment of swap
14287 <1> ; core - beginning of process program
14288 <1> ; ecore - end of core
14289 <1> ; user - start of user parameter area
14290 <1> ; u.uno - user process number
14291 <1> ; p.dska - holds block number of process
14292 <1> ; OUTPUTS ->
14293 <1> ; swp I/O queue
14294 <1> ; p.break - negative word count of process
14295 <1> ; r1 - process disk address
14296 <1> ; r2 - negative word count
14297 <1> ;
14298 <1> ; RETRO UNIX 8086 v1 input/output:
14299 <1> ;
14300 <1> ; INPUTS ->
14301 <1> ; u.uno - process number (to be swapped out)
14302 <1> ; OUTPUTS ->
14303 <1> ; none
14304 <1> ;
14305 <1> ; ((Modified registers: ECX, ESI, EDI))
14306 <1> ;
14307 <1> ;
14308 <1> ; 28/02/2017
14309 <1> ;cmp byte [multi_tasking], 0 ; Musti tasking mode ?
14310 <1> ;jna short wswap
14311 000109C7 803D[D38E0100]00 <1> cmp byte [u.fpsave], 0 ; 28/02/2017
14312 000109CE 7606 <1> jna short wswap
14313 000109D0 DD35[D48E0100] <1> fnsave [u.fpregs] ; save floating point registers (94 bytes)
14314 <1> wswap:
14315 000109D6 8B3D[A88E0100] <1> mov edi, [u.upage] ; process's user (u) structure page addr
14316 000109DC B93D000000 <1> mov ecx, (U_SIZE + 3) / 4
14317 000109E1 BE[4C8E0100] <1> mov esi, user ; active user (u) structure
14318 000109E6 F3A5 <1> rep movsd
14319 <1> ;
14320 000109E8 8B35[508E0100] <1> mov esi, [u.usp] ; esp (system stack pointer,
14321 <1> ; points to user registers)
14322 000109EE 8B0D[4C8E0100] <1> mov ecx, [u.sp] ; return address from the system call
14323 <1> ; (for IRET)
14324 <1> ; [u.sp] -> EIP (user)
14325 <1> ; [u.sp+4]-> CS (user)
14326 <1> ; [u.sp+8] -> EFLAGS (user)
14327 <1> ; [u.sp+12] -> ESP (user)
14328 <1> ; [u.sp+16] -> SS (user)
14329 000109F4 29F1 <1> sub ecx, esi ; required space for user registers
14330 000109F6 83C114 <1> add ecx, 20 ; +5 dwords to return from system call
14331 <1> ; (for IRET)
14332 000109F9 C1E902 <1> shr ecx, 2
14333 000109FC F3A5 <1> rep movsd
14334 000109FE C3 <1> retn
14335 <1>
14336 <1> rswap: ; < swap in, swap from disk >
14337 <1> ; 28/02/2017 (frstor)
14338 <1> ; 15/01/2017
14339 <1> ; 14/01/2017
14340 <1> ; 21/05/2016
14341 <1> ; 03/05/2016
14342 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)

```

```

14343 <1> ; 09/05/2015 - 15/09/2015 (Retro UNIX 386 v1)
14344 <1> ; 26/05/2013 - 08/03/2014 (Retro UNIX 8086 v1)
14345 <1> ; 'rswap' reads a process whose number is in r1,
14346 <1> ; from disk into core.
14347 <1>
14348 <1> ; Retro UNIX 386 v1 modification ->
14349 <1> ; User (u) structure content and the user's register content
14350 <1> ; will be restored from process's/user's UPAGE (a page for
14351 <1> ; saving 'u' structure and user registers for task switching).
14352 <1> ; u.usp - points to kernel stack address which contains
14353 <1> ; user's registers while entering system call.
14354 <1> ; u.sp - points to kernel stack address
14355 <1> ; to return from system call -for IRET-.
14356 <1> ; [u.usp]+32+16 = [u.sp]
14357 <1> ; [u.usp] -> edi, esi, ebp, esp (= [u.usp]+32), ebx,
14358 <1> ; edx, ecx, eax, gs, fs, es, ds, -> [u.sp].
14359 <1>
14360 <1> ; RETRO UNIX 8086 v1 modification ->
14361 <1> ; 'swap to disk' is replaced with 'change running segment'
14362 <1> ; according to 8086 cpu (x86 real mode) architecture.
14363 <1> ; pdp-11 was using 64KB uniform memory while IBM PC
14364 <1> ; compatibles was using 1MB segmented memory
14365 <1> ; in 8086/8088 times.
14366 <1>
14367 <1> ; INPUTS ->
14368 <1> ; r1 - process number of process to be read in
14369 <1> ; p.break - negative of word count of process
14370 <1> ; p.dska - disk address of the process
14371 <1> ; u.emt - determines handling of emt's
14372 <1> ; u.ilgins - determines handling of illegal instructions
14373 <1> ; OUTPUTS ->
14374 <1> ; 8 = (u.ilgins)
14375 <1> ; 24 = (u.emt)
14376 <1> ; swp - bit 10 is set to indicate read
14377 <1> ; (bit 15=0 when reading is done)
14378 <1> ; swp+2 - disk block address
14379 <1> ; swp+4 - negative word count
14380 <1> ; ((swp+6 - address of user structure))
14381 <1>
14382 <1> ; RETRO UNIX 8086 v1 input/output:
14383 <1>
14384 <1> ; INPUTS ->
14385 <1> ; AL - new process number (to be swapped in)
14386 <1> ; OUTPUTS ->
14387 <1> ; none
14388 <1>
14389 <1> ; ((Modified registers: EAX, ECX, ESI, EDI, ESP))
14390 <1>
14391 <1> ; Retro UNIX 386 v1 - modification ! 14/05/2015
14392 <1> mov esi, eax ; process's user (u) structure page addr
14393 <1> mov ecx, (U_SIZE + 3) / 4
14394 <1> mov edi, user ; active user (u) structure
14395 <1> rep movsd
14396 <1> pop eax ; 'rswap' return address
14397 <1>
14398 <1> ; cli
14399 <1> mov edi, [u.usp] ; esp (system stack pointer,
14400 <1> ; points to user registers)
14401 <1> mov esp, edi ; 14/01/2017
14402 <1> mov ecx, [u.sp] ; return address from the system call
14403 <1> ; (for IRET)
14404 <1> ; [u.sp] -> EIP (user)
14405 <1> ; [u.sp+4] -> CS (user)
14406 <1> ; [u.sp+8] -> EFLAGS (user)
14407 <1> ; [u.sp+12] -> ESP (user)
14408 <1> ; [u.sp+16] -> SS (user)
14409 <1> sub ecx, edi ; required space for user registers
14410 <1> add ecx, 20 ; +5 dwords to return from system call
14411 <1> ; (for IRET)
14412 <1> shr ecx, 2
14413 <1> rep movsd
14414 <1> ; mov esp, [u.usp] ; 15/09/2015
14415 <1> ; sti
14416 <1> ; 28/02/2017
14417 <1> ; cmp byte [multi_tasking], 0 ; Multi tasking mode ?
14418 <1> ; jna short rswap_retn
14419 <1> cmp byte [u.fpsave], 0
14420 <1> jna short rswap_retn
14421 <1> frstor [u.fpregs] ; restore floating point regs (94 bytes)
14422 <1> ; 108 bytes (22/08/2024)
14423 <1> rswap_retn:
14424 <1> push eax ; 'rswap' return address
14425 <1> retn
14426 <1>
14427 <1> putlu:
14428 <1> ; 20/05/2016
14429 <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
14430 <1> ; 10/05/2015 - 12/09/2015 (Retro UNIX 386 v1)
14431 <1> ; 15/04/2013 - 23/02/2014 (Retro UNIX 8086 v1)
14432 <1> ; 'putlu' is called with a process number in r1 and a pointer
14433 <1> ; to lowest priority Q (runq+4) in r2. A link is created from
14434 <1> ; the last process on the queue to process in r1 by putting
14435 <1> ; the process number in r1 into the last process's link.
14436 <1>
14437 <1> ; INPUTS ->
14438 <1> ; r1 - user process number
14439 <1> ; r2 - points to lowest priority queue
14440 <1> ; p.dska - disk address of the process
14441 <1> ; u.emt - determines handling of emt's
14442 <1> ; u.ilgins - determines handling of illegal instructions
14443 <1> ; OUTPUTS ->
14444 <1> ; r3 - process number of last process on the queue upon
14445 <1> ; entering putlu
14446 <1> ; p.link-1 + r3 - process number in r1
14447 <1> ; r2 - points to lowest priority queue
14448 <1>
14449 <1> ; ((Modified registers: EDX, EBX))
14450 <1>
14451 <1> ; / r1 = user process no.; r2 points to lowest priority queue
14452 <1>
14453 <1> ; EBX = r2
14454 <1> ; EAX = r1 (AL=r1b)
14455 <1>
14456 <1> ; 20/05/2016
14457 <1> ; AL = process number (1 to 16) // Retro UNIX 8086, 386 v1 //
14458 <1> ; (max. 16 processes available for current kernel version)
14459 <1> ; EBX = run queue address ; 20/05/2016 (TRDOS 386)
14460 <1> ; which is one of following addresses:
14461 <1> ; 1) 'runq_event' high priority run queue
14462 <1> ; 2) 'runq_normal' normal/regular priority run queue
14463 <1> ; 3) 'runq_background' low priority run queue
14464 <1>
14465 <1> ; mov ebx, runq
14466 <1> movzx edx, byte [ebx]

```

```

14467 00010A3A 43      <1>      inc     ebx
14468 00010A3B 20D2    <1>      and     dl, dl
14469                <1>      ; tstb (r2)+ / is queue empty?
14470 00010A3D 740A    <1>      jz      short putlu_1
14471                <1>      ; beg 1f / yes, branch
14472 00010A3F 8A13    <1>      mov     dl, [ebx] ; 12/09/2015
14473                <1>      ; movb (r2), r3 / no, save the "last user" process number
14474                <1>      ; / in r3
14475 00010A41 8882[778D0100] <1>      mov     [edx+p.link-1], al
14476                <1>      ; movb r1, p.link-1(r3) / put pointer to user on
14477                <1>      ; / "last users" link
14478 00010A47 EB03    <1>      jmp     short putlu_2
14479                <1>      ; br 2f /
14480                <1>      putlu_1: ; 1:
14481 00010A49 8843FF    <1>      mov     [ebx-1], al
14482                <1>      ; movb r1, -1(r2) / user is only user;
14483                <1>      ; / put process no. at beginning and at end
14484                <1>      putlu_2: ; 2:
14485 00010A4C 8803    <1>      mov     [ebx], al
14486                <1>      ; movb r1, (r2) / user process in r1 is now the last entry
14487                <1>      ; / on the queue
14488 00010A4E 88C2    <1>      mov     dl, al
14489 00010A50 88B2[778D0100] <1>      mov     [edx+p.link-1], dh ; 0
14490                <1>      ; dec r2 / restore r2
14491 00010A56 C3        <1>      retn
14492                <1>      ; rts r0
14493                <1>
14494                <1>      sysver:
14495                <1>      ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
14496 00010A57 C705[548E0100]0002- <1>      mov     dword [u.r0], 200h ; AH = major version, AL = minor version
14497 00010A5F 0000    <1>
14498                <1>      jmp     sysret
14499                <1>
14500                <1>      syspri: ; change running priority (of the process)
14501                <1>      ; 23/07/2022 - TRDOS 386 v2.0.5
14502                <1>      ; 21/05/2016
14503                <1>      ; 20/05/2026 - TRDOS 386 (TRDOS v2.0)
14504                <1>      ; INPUT ->
14505                <1>      ; BL = priority level
14506                <1>      ; 0 = low running priority (running on background)
14507                <1>      ; 1 = normal/regular priority (running as regular)
14508                <1>      ; 2 = high/event priority (running for event)
14509                <1>      ; >2 = invalid, it will accepted as 2 (event)
14510                <1>      ; 0FFh = get/return current running priority only
14511                <1>      ; OUTPUT ->
14512                <1>      ; * if current [u.pri] < 2
14513                <1>      ; if BL input < 0FFh ->
14514                <1>      ; [u.pri] is updated as in BL input (0,1,2)
14515                <1>      ; if BL input = 0FFh -> AL = [u.pri] (current)
14516                <1>      ;
14517                <1>      ; * if current [u.pri] = 2
14518                <1>      ; if BL input < 0FFh -> cf = 1 & AL = 2
14519                <1>      ; if BL input = 0FFh -> cf = 0 & AL = 2
14520                <1>      ;
14521                <1>      ; NOTE:
14522                <1>      ; If [u.pri] = 2, it can not be changed to 1 or 0;
14523                <1>      ; because, run queue of the running process is unspecified
14524                <1>      ; at this stage. Process might be started by a timer event
14525                <1>      ; or priority might be changed to high by previous
14526                <1>      ; 'syspri' system call. In both cases, the process is in
14527                <1>      ; 'runq_normal' or 'runq_background' queue.
14528                <1>      ; As result of this fact, when the [u.quant] time quantum
14529                <1>      ; of the process is elapsed or 'sysrele' system call is
14530                <1>      ; instructed by the process, 'tswap' ('tswitch') procedure
14531                <1>      ; will be called (to 'swap' or 'switch' out the procedure)
14532                <1>      ; and it will not call 'putlu' to add the (stopping)
14533                <1>      ; process to relevant run queue when [u.pri] = 2.
14534                <1>      ; (Otherwise, it would be possible to add process to
14535                <1>      ; a run queue while it is already in a run queue, wrongly.)
14536                <1>      ;
14537                <1>      ; If [u.pri] < 2, 'tswap/tswitch' procedure will call
14538                <1>      ; 'putlu' to add process to relevant run queue
14539                <1>      ; according to [u.pri] value. ('runq_normal' for 1,
14540                <1>      ; 'runq_background' for 0).
14541                <1>      ;
14542                <1>      ; If BL input >= 2 and < 0FFh while [u.pri] < 2,
14543                <1>      ; process will be added to 'runq_normal' queue and
14544                <1>      ; [u.pri] will be set to 2. (in 'syspri' system call)
14545                <1>      ;
14546 00010A66 29C0    <1>      sub     eax, eax ; 0
14547 00010A68 A3[C08E0100] <1>      mov     [u.error], eax
14548                <1>
14549 00010A6D A0[9E8E0100] <1>      mov     al, [u.pri]
14550 00010A72 A3[548E0100] <1>      mov     [u.r0], eax
14551                <1>
14552 00010A77 FEC3    <1>      inc     bl
14553                <1>      ; jz      sysret ; 0FFh -> 0, get priority level
14554                <1>      ; 23/07/2022
14555 00010A79 742C    <1>      jz      short syspri_2 ; jmp sysret
14556                <1>
14557 00010A7B 3C02    <1>      cmp     al, 2
14558                <1>      ; jnb     error ; CF = 1 & AL = 2 (& last error = 0)
14559                <1>      ; 23/07/2022
14560 00010A7D 7205    <1>      jb      short syspri_0
14561 00010A7F E9DABFFFFF <1>      jmp     error
14562                <1>      syspri_0:
14563 00010A84 FECB    <1>      dec     bl
14564 00010A86 80FB02    <1>      cmp     bl, 2
14565 00010A89 7602    <1>      jna     short syspri_1
14566 00010A8B B302    <1>      mov     bl, 2
14567                <1>      syspri_1:
14568 00010A8D 881D[9E8E0100] <1>      mov     [u.pri], bl
14569 00010A93 80FB02    <1>      cmp     bl, 2
14570                <1>      ; jnb     sysret
14571                <1>      ; 23/07/2022
14572 00010A96 720F    <1>      jb      short syspri_2 ; jmp sysret
14573                <1>
14574                <1>      ; here...
14575                <1>      ; Priority of current process has been changed to high
14576                <1>      ; ('run for event') but current process will be added to
14577                <1>      ; 'run as normal' queue. ('run for event' high priority
14578                <1>      ; queue is under control of timer -& RTC- interrupt only!)
14579                <1>      ;
14580                <1>      ; (Otherwise, process can fall into black hole!
14581                <1>      ; e.g. if it is not in waiting list and it has not got
14582                <1>      ; a timer event and it is not in a run queue!
14583                <1>      ; Because, when [u.pri] is 2, 'tswap/tswitch' will not
14584                <1>      ; add the stopping process to a run queue.)
14585                <1>
14586 00010A98 A0[A58E0100] <1>      mov     al, [u.uno]
14587 00010A9D BB[448E0100] <1>      mov     ebx, runq_normal ; normal priority !
14588                <1>      ; [u.pri] is set to high
14589                <1>      ; but 'runq_event' queue is set

```

```

14590                                     <1>                                     ; only by the kernel's timer
14591                                     <1>                                     ; event function (timer interrupt).
14592 00010AA2 E890FFFFFF                                     <1>                                     call    putlu
14593                                     <1> syspri_2:                                     <1>
14594 00010AA7 E9D2BFFFFF                                     <1>                                     jmp     sysret
14595                                     <1>
14596                                     <1> cpass: ; / get next character from user area of core and put it in AL (r1)
14597                                     <1> ; 30/07/2022 - TRDOS 386 Kernel v2.0.5
14598                                     <1> ; 02/05/2016 - TRDOS 386 (TRDOS v2.0)
14599                                     <1> ; 19/05/2015 - 18/10/2015 (Retro UNIX 386 v1)
14600                                     <1> ; 14/08/2013 - 20/09/2013 (Retro UNIX 8086 v1)
14601                                     <1> ; INPUTS ->
14602                                     <1> ; [u.base] = virtual address in user area
14603                                     <1> ; [u.count] = byte count (max.)
14604                                     <1> ; [u.pcount] = byte count in page (0 = reset)
14605                                     <1> ; OUTPUTS ->
14606                                     <1> ; AL = the character which is pointed by [u.base]
14607                                     <1> ; zf = 1 -> transfer count has been completed
14608                                     <1> ;
14609                                     <1> ; ((Modified registers: EAX, EDX, ECX))
14610                                     <1>
14611                                     <1> ; 30/07/2022
14612 00010AAC 29C0                                     <1> sub     eax, eax
14613                                     <1>
14614 00010AAE 3905[7C8E0100] <1> cmp     [u.count], eax ; 0
14615                                     <1> ; cmp     dword [u.count], 0 ; have all the characters been transferred
14616                                     <1> ; ; i.e., u.count, # of chars. left
14617 00010AB4 763D                                     <1> jna     short cpass_3 ; to be transferred = 0?) yes, branch
14618 00010AB6 FF0D[7C8E0100] <1> dec     dword [u.count] ; no, decrement u.count
14619                                     <1> ; 19/05/2015
14620                                     <1> ; (Retro UNIX 386 v1 - translation from user's virtual address
14621                                     <1> ; to physical address
14622                                     <1> ; 30/07/2022
14623 00010ABC 663905[B88E0100] <1> cmp     [u.pcount], ax ; 0
14624                                     <1> ; cmp     word [u.pcount], 0 ; byte count in page = 0 (initial value)
14625                                     <1> ; ; 1-4095 --> use previous physical base address
14626                                     <1> ; ; in [u.pbbase]
14627 00010AC3 770D                                     <1> ja      short cpass_1
14628                                     <1> ; 30/07/2022
14629 00010AC5 3905[B08E0100] <1> cmp     [u.ppgdir], eax ; 0
14630                                     <1> ; cmp     dword [u.ppgdir], 0 ; is the caller os kernel
14631 00010ACB 7427                                     <1> je      short cpass_k ; (sysexec, '/etc/init') ? (MainProg)
14632 00010ACD E861FDFFFF <1> call    trans_addr_r
14633                                     <1> cpass_1:
14634 00010AD2 66FF0D[B88E0100] <1> dec     word [u.pcount]
14635                                     <1> cpass_2:
14636 00010AD9 8B15[B48E0100] <1> mov     edx, [u.pbbase]
14637 00010ADF 8A02                                     <1> mov     al, [edx] ; take the character pointed to
14638                                     <1> ; by u.base and put it in r1
14639 00010AE1 FF05[808E0100] <1> inc     dword [u.nread] ; increment no. of bytes transferred
14640 00010AE7 FF05[788E0100] <1> inc     dword [u.base] ; increment the buffer address to point to the
14641                                     <1> ; next byte
14642 00010AED FF05[B48E0100] <1> inc     dword [u.pbbase]
14643                                     <1> cpass_3:
14644 00010AF3 C3                                     <1> retn
14645                                     <1> cpass_k:
14646                                     <1> ; 02/07/2015
14647                                     <1> ; The caller is os kernel
14648                                     <1> ; (get sysexec arguments from kernel's memory space)
14649 00010AF4 8B1D[788E0100] <1> mov     ebx, [u.base]
14650 00010AFA 66C705[B88E0100]00- <1> mov     word [u.pcount], PAGE_SIZE ; 4096
14650 00010B02 10                                     <1>
14651 00010B03 891D[B48E0100] <1> mov     [u.pbbase], ebx
14652 00010B09 EBCE                                     <1> jmp     short cpass_2
14653                                     <1>
14654                                     <1> transfer_to_user_buffer: ; fast transfer
14655                                     <1> ; 27/05/2016
14656                                     <1> ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
14657                                     <1> ;
14658                                     <1> ; INPUT ->
14659                                     <1> ; ESI = source address in system space
14660                                     <1> ; EDI = user's buffer address
14661                                     <1> ; ECX = transfer (byte) count
14662                                     <1> ; [u.pgdir] = user's page directory
14663                                     <1> ; OUTPUT ->
14664                                     <1> ; ECX = actual transfer count
14665                                     <1> ; cf = 1 -> error
14666                                     <1> ; [u.count] = remain byte count
14667                                     <1> ;
14668                                     <1> ; Modified registers: eax, ecx
14669                                     <1> ;
14670                                     <1> ;
14671 00010B0B 21C9                                     <1> and     ecx, ecx
14672 00010B0D 743B                                     <1> jz      short ttub_4
14673                                     <1>
14674 00010B0F 890D[7C8E0100] <1> mov     [u.count], ecx
14675                                     <1>
14676 00010B15 57                                     <1> push    edi
14677 00010B16 56                                     <1> push    esi
14678 00010B17 53                                     <1> push    ebx
14679 00010B18 52                                     <1> push    edx
14680 00010B19 51                                     <1> push    ecx
14681                                     <1>
14682 00010B1A 89FB                                     <1> mov     ebx, edi
14683 00010B1C 81C300004000 <1> add     ebx, CORE ; 27/05/2016
14684                                     <1> ttub_1:
14685                                     <1> ; ebx = virtual (linear) address
14686                                     <1> ; [u.pgdir] = user's page directory
14687 00010B22 E8E350FFFF <1> call    get_physical_addr_x ; get physical address
14688 00010B27 7222                                     <1> jc      short ttub_5
14689                                     <1> ; eax = physical address
14690                                     <1> ; ecx = remain byte count in page (1-4096)
14691 00010B29 89C7                                     <1> mov     edi, eax
14692 00010B2B A1[7C8E0100] <1> mov     eax, [u.count]
14693 00010B30 39C1                                     <1> cmp     ecx, eax
14694 00010B32 7602                                     <1> jna     short ttub_2
14695 00010B34 89C1                                     <1> mov     ecx, eax
14696                                     <1> ttub_2:
14697 00010B36 29C8                                     <1> sub     eax, ecx
14698 00010B38 01CB                                     <1> add     ebx, ecx
14699 00010B3A F3A4                                     <1> rep     movsb
14700 00010B3C A3[7C8E0100] <1> mov     [u.count], eax
14701 00010B41 09C0                                     <1> or      eax, eax
14702 00010B43 75DD                                     <1> jnz     short ttub_1
14703                                     <1> ttub_retn:
14704                                     <1> tfub_retn:
14705 00010B45 59                                     <1> pop     ecx ; transfer count = actual transfer count
14706                                     <1> ttub_3:
14707 00010B46 5A                                     <1> pop     edx
14708 00010B47 5B                                     <1> pop     ebx
14709 00010B48 5E                                     <1> pop     esi
14710 00010B49 5F                                     <1> pop     edi
14711                                     <1> ttub_4:
14712 00010B4A C3                                     <1> retn

```

```

14713      <1> ttub_5:
14714 00010B4B 59      <1>      pop      ecx
14715 00010B4C 2B0D[7C8E0100] <1>      sub      ecx, [u.count] ; actual transfer count
14716 00010B52 F9      <1>      stc
14717 00010B53 EBF1    <1>      jmp      short ttub_3
14718      <1>
14719      <1> transfer_from_user_buffer: ; fast transfer
14720      <1>      ; 27/05/2016
14721      <1>      ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
14722      <1>      ;
14723      <1>      ; INPUT ->
14724      <1>      ;     ESI = user's buffer address
14725      <1>      ;     EDI = destination address in system space
14726      <1>      ;     ECX = transfer (byte) count
14727      <1>      ;     [u.pgdir] = user's page directory
14728      <1>      ; OUTPUT ->
14729      <1>      ;     ecx = actual transfer count
14730      <1>      ;     cf = 1 -> error
14731      <1>      ;     [u.count] = remain byte count
14732      <1>      ;
14733      <1>      ; Modified registers: eax, ecx
14734      <1>      ;
14735      <1>
14736 00010B55 21C9    <1>      and      ecx, ecx
14737      <1>      ;jz      short tfub_4
14738 00010B57 74F1    <1>      jz      short ttub_4
14739      <1>
14740 00010B59 890D[7C8E0100] <1>      mov      [u.count], ecx
14741      <1>
14742 00010B5F 57      <1>      push     edi
14743 00010B60 56      <1>      push     esi
14744 00010B61 53      <1>      push     ebx
14745 00010B62 52      <1>      push     edx
14746 00010B63 51      <1>      push     ecx
14747      <1>
14748 00010B64 89F3    <1>      mov      ebx, esi
14749 00010B66 81C300004000 <1>      add      ebx, CORE ; 27/05/2016
14750      <1> tfub_1:
14751      <1>      ; ebx = virtual (linear) address
14752      <1>      ; [u.pgdir] = user's page directory
14753 00010B6C E89950FFFF <1>      call     get_physical_addr_x ; get physical address
14754      <1>      ;jc      short tfub_5
14755 00010B71 72D8    <1>      jc      short ttub_5
14756      <1>      ; eax = physical address
14757      <1>      ; ecx = remain byte count in page (1-4096)
14758 00010B73 89C6    <1>      mov      esi, eax
14759 00010B75 A1[7C8E0100] <1>      mov      eax, [u.count]
14760 00010B7A 39C1    <1>      cmp      ecx, eax
14761 00010B7C 7602    <1>      jna      short tfub_2
14762 00010B7E 89C1    <1>      mov      ecx, eax
14763      <1> tfub_2:
14764 00010B80 29C8    <1>      sub      eax, ecx
14765 00010B82 01CB    <1>      add      ebx, ecx
14766 00010B84 F3A4    <1>      rep      movsb
14767 00010B86 A3[7C8E0100] <1>      mov      [u.count], eax
14768 00010B8B 09C0    <1>      or       eax, eax
14769 00010B8D 75DD    <1>      jnz      short tfub_1
14770      <1>
14771 00010B8F EBB4    <1>      jmp      short tfub_retn
14772      <1>
14773      <1> ;tfub_retn:
14774      <1>      pop      ecx ; transfer count = actual transfer count
14775      <1> ;tfub_3:
14776      <1>      pop      edx
14777      <1>      pop      ebx
14778      <1>      pop      esi
14779      <1>      pop      edi
14780      <1> ;tfub_4:
14781      <1>      ; retn
14782      <1> ;tfub_5:
14783      <1>      pop      ecx
14784      <1>      sub      ecx, [u.count] ; actual transfer count
14785      <1>      stc
14786      <1>      jmp      short tfub_3
14787      <1>
14788      <1> sysfff: ; <Find First File>
14789      <1>      ; 25/08/2024 (TRDOS 386 kernel v2.0.9)
14790      <1>      ; 08/08/2022
14791      <1>      ; 30/07/2022 (TRDOS 386 kernel v2.0.5)
14792      <1>      ; 17/10/2016
14793      <1>      ; 16/10/2016
14794      <1>      ; 15/10/2016 TRDOS 386 (TRDOS v2.0) feature only !
14795      <1>      ;
14796      <1>      ; -derived from TRDOS v1.0, INT_21H.ASM-
14797      <1>      ; ("loc_INT21h_find_first_file")
14798      <1>      ; TRDOS 8086 (v1.0)
14799      <1>      ; 07/08/2011
14800      <1>      ; Find First File
14801      <1>      ; INPUT:
14802      <1>      ;     CX= Attributes
14803      <1>      ;     DS:DX= Pointer to filename
14804      <1>      ; MSDOS OUTPUT:
14805      <1>      ;     DTA: (Default address: PSP offset 80h)
14806      <1>      ;     Offset Description
14807      <1>      ;     0 Reserved for use find next file
14808      <1>      ;     21 Attribute of file found
14809      <1>      ;     22 Time stamp of file
14810      <1>      ;     24 Date stamp of file
14811      <1>      ;     26 File size in bytes
14812      <1>      ;     30 Filename and extension (zero terminated)
14813      <1>      ; If cf = 1:
14814      <1>      ;     Error Codes: (in AX)
14815      <1>      ;     2 - File not found
14816      <1>      ;     18 - No more files
14817      <1>      ;
14818      <1>      ; TRDOS 386 (v2.0)
14819      <1>      ; 15/10/2016
14820      <1>      ;
14821      <1>      ; INPUT ->
14822      <1>      ;     CL = File attributes
14823      <1>      ;     bit 0 (1) - Read only file (R)
14824      <1>      ;     bit 1 (1) - Hidden file (H)
14825      <1>      ;     bit 2 (1) - System file (R)
14826      <1>      ;     bit 3 (1) - Volume label/name (V)
14827      <1>      ;     bit 4 (1) - Subdirectory (D)
14828      <1>      ;     bit 5 (1) - File has been archived (A)
14829      <1>      ;     CH = 0 -> Return basic parameters (24 bytes)
14830      <1>      ;     CH > 0 -> Return FindFile structure/table (128 bytes)
14831      <1>      ;     EBX = Pointer to filename (ASCIIIZ) -path-
14832      <1>      ;     EDX = File parameters buffer address
14833      <1>      ;     (buffer size = 24 bytes if CH input = 0)
14834      <1>      ;     (buffer size = 128 bytes if CH input > 0)
14835      <1>      ; OUTPUT ->
14836      <1>      ;     EAX = 0 if CH input > 0

```

```

14837 <1> ; EAX = First cluster number of file if CH input = 0
14838 <1> ; EDX = File parameters table/structure address
14839 <1> ; Basic Parameters:
14840 <1> ; Offset Description
14841 <1> ; -----
14842 <1> ; 0 File Attributes
14843 <1> ; 1 Ambiguous filename chars are used sign
14844 <1> ; (0 = filename fits exactly with request)
14845 <1> ; (>0 = ambiguous filename chars are used)
14846 <1> ; 2 Time stamp of file
14847 <1> ; 4 Date stamp of file
14848 <1> ; 6 File size in bytes
14849 <1> ; 10 Short Filename (ASCIIIZ, max. 13 bytes)
14850 <1> ; 23 Longname Length (1-255) if existing
14851 <1> ;
14852 <1> ; cf = 1 -> Error code in AL
14853 <1> ;
14854 <1> ; Modified Registers: EAX (at the return of system call)
14855 <1> ;
14856 <1> ; TR-DOS FindFile (FFF) Structure (128 bytes):
14857 <1> ; 09/10/2011 (DIR.ASM) - 10/02/2016 (trdoskx.s)
14858 <1> ;
14859 <1> ; Offset Parameter Size
14860 <1> ; -----
14861 <1> ; 0 FindFile_Drv 1 byte
14862 <1> ; 1 FindFile_Directory 65 bytes
14863 <1> ; 66 FindFile_Name 13 bytes
14864 <1> ; 79 FindFile_LongNameEntryLength 1 byte
14865 <1> ; Above 80 bytes form
14866 <1> ; TR-DOS Source/Destination File FullName Format/Structure
14867 <1> ; 80 FindFile_AttributesMask 1 word
14868 <1> ; 82 FindFile_DirEntry 32 bytes (*)
14869 <1> ; 114 FindFile_DirFirstCluster 1 double word
14870 <1> ; 118 FindFile_DirCluster 1 double word
14871 <1> ; 122 FindFile_DirEntryNumber 1 word
14872 <1> ; 124 FindFile_MatchCounter 1 word
14873 <1> ; 126 FindFile_Reserved 1 word
14874 <1> ; (*) MS-DOS, FAT 12-16-32 classic directory entry (32 bytes)
14875 <1> ;
14876 <1> ; mov [u.name], ebx
14877 <1> ; 16/10/2016
14878 00010B91 8915[54830100] <1> mov [FFF_UBuffer], edx
14879 00010B97 66890D[59830100] <1> mov [FFF_Attrib], cx ; [FFF_RType] = ch
14880 <1> ; Attributes in CL, return data type in CH
14881 00010B9E 89DE <1> mov esi, ebx
14882 <1> ; file name is forced, change directory as temporary
14883 <1> ; mov ax, 1
14884 <1> ; mov [FFF_Valid], ah ; 0 ; reset ; 17/10/2016
14885 <1> ; call set_working_path
14886 00010BA0 E820110000 <1> call set_working_path_x ; 17/10/2016
14887 00010BA5 731D <1> jnc short sysfff_0
14888 <1> ;
14889 00010BA7 21C0 <1> and eax, eax ; 0 -> Bad Path!
14890 00010BA9 7505 <1> jnz short sysfff_err
14891 <1> ;
14892 <1> ; eax = 0
14893 00010BAB B80C000000 <1> mov eax, ERR_DIR_NOT_FOUND ; Directory not found !
14894 <1> sysfff_err:
14895 <1> mov [u.r0], eax
14896 00010BB5 A3[C08E0100] <1> mov [u.error], eax
14897 00010BBA E8DB110000 <1> call reset_working_path
14898 00010BBF E99ABEFFFF <1> jmp error
14899 <1> ;
14900 <1> sysfff_0:
14901 <1> ;;;
14902 <1> ; 25/08/2024 (bugfix)
14903 <1> ; mov al, [esp] ; ???
14904 00010BC4 A0[59830100] <1> mov al, [FFF_Attrib]
14905 <1> ;;;
14906 00010BC9 08C0 <1> or al, al
14907 00010BCB 7412 <1> jz short sysfff_2
14908 00010BCD B410 <1> mov ah, 10h
14909 00010BCF A808 <1> test al, 08h
14910 00010BD1 7503 <1> jnz short sysfff_1
14911 00010BD3 80CC08 <1> or ah, 08h
14912 <1> sysfff_1:
14913 00010BD6 2410 <1> and al, 10h ; Directory
14914 00010BD8 7405 <1> jz short sysfff_2
14915 00010BDA 80E408 <1> and ah, 08h
14916 00010BDD 30C0 <1> xor al, al ; When a directory is searched,
14917 <1> ; filename will be returned even if
14918 <1> ; it is not a directory!
14919 <1> ; Because: (in order to prevent
14920 <1> ; creating a dir with existing file name)
14921 <1> ; Dir and file names must not be same!
14922 <1> ; (return attribute must be checked)
14923 <1> sysfff_2:
14924 <1> ; AX = Attributes mask
14925 <1> ; AL = AND mask (result must be equal to AL)
14926 <1> ; AH = Negative AND mask (result must be ZERO)
14927 <1> ; ESI = FindFile_Name address
14928 <1> ;
14929 00010BDF E8867DFFFF <1> call find_first_file
14930 00010BE4 72CA <1> jc short sysfff_err ; eax = 2 (File not found !)
14931 <1> ;
14932 <1> ; ESI = Directory Entry (FindFile_DirEntry) Location
14933 <1> ; EDI = Directory Buffer Directory Entry Location
14934 <1> ; EAX = File Size
14935 <1> ; BL = Attributes of The File/Directory
14936 <1> ; BH = Long Name Yes/No Status (>0 is YES)
14937 <1> ; DX > 0 : Ambiguous filename chars are used
14938 <1> ;
14939 <1> sysfff_3:
14940 <1> ; 16/10/2016
14941 00010BE6 668B0D[59830100] <1> mov cx, [FFF_Attrib]
14942 <1> ; Attribs in CL, return data type in CH
14943 <1> ;
14944 <1> ; or cl, cl
14945 <1> ; jz short sysfff_4 ; 0 = No filter
14946 00010BED 80F1FF <1> xor cl, 0FFh
14947 <1> ; cl = negative attributes ; 25/08/2024
14948 00010BF0 20D9 <1> and cl, bl
14949 00010BF2 7409 <1> jz short sysfff_4
14950 <1> ;
14951 <1> ; mov eax, 2 ; 'file not found !' error
14952 <1> ; jmp short sysfff_err_1
14953 <1> ;
14954 <1> ; 16/10/2016
14955 00010BF4 E81E7EFFFF <1> call find_next_file
14956 00010BF9 72B5 <1> jc short sysfff_err ; eax = 12 (no more files !)
14957 00010BFB EBE9 <1> jmp short sysfff_3
14958 <1> ;
14959 <1> sysfff_4:
14960 00010BFD 20ED <1> and ch, ch ; [FFF_RType]

```



```

14961      <1>      ;jz      short sysfff_5
14962      <1>      ; 25/08/2024
14963 00010BFF 755A      <1>      jnz      short sysfff_7
14964      <1>
14965      <1>      ;mov     ecx, 128 ; ; transfer length
14966      <1>      ; 30/07/2022
14967      <1>      ;sub     ecx, ecx
14968      <1>      ;mov     cl, 128
14969      <1>      ;mov     [FFF_Valid], cl
14970      <1>      ; 25/08/2024
14971      <1>      ;mov     byte [FFF_Valid], 128 ; (*)
14972      <1>      ;jmp     short sysfff_7 ; sysfnf_11
14973      <1>
14974      <1> sysfff_5:
14975      <1>      ;mov     esi, FindFile_DirEntry
14976      <1>      ;mov     ecx, 24 ; transfer length
14977      <1>      ; 30/07/2022
14978      <1>      ;sub     ecx, ecx
14979      <1>      ;mov     cl, 24
14980      <1>      ;mov     [FFF_Valid], cl
14981 00010C01 C605[58830100]18      <1>      mov     byte [FFF_Valid], 24 ; (*)
14982      <1> sysfnf_12:
14983 00010C08 BF[24880100]      <1>      mov     edi, DTA ; FFF data transfer address
14984      <1>      ;mov     al, [esi+DirEntry_Attr] ; 11
14985      <1>      mov     al, bl ; File/Dir Attributes
14986 00010C0F 887F17      <1>      mov     [edi+23], bh ; Longname length (0= none)
14987 00010C12 AA      <1>      stosb
14988 00010C13 88D0      <1>      mov     al, dl ; DL is for '?'
14989 00010C15 00F0      <1>      add     al, dh ; DH is for '*'
14990      <1>      ; AL > 0 if ambiguous file name wildcards are used
14991 00010C17 AA      <1>      stosb
14992 00010C18 8B4616      <1>      mov     eax, [esi+DirEntry_WrtTime] ; 22
14993 00010C1B AB      <1>      stosd ; DirEntry_WrtTime & DirEntry_WrtDate
14994 00010C1C 8B461C      <1>      mov     eax, [esi+DirEntry_FileSize] ; 28
14995 00010C1F AB      <1>      stosd
14996 00010C20 668B4614      <1>      mov     ax, [esi+DirEntry_FstClusHI] ; 20
14997      <1>      ;shl     ax, 16
14998      <1>      ; 23/07/2022 (BugFix)
14999 00010C24 C1E010      <1>      shl     eax, 16
15000 00010C27 668B461A      <1>      mov     ax, [esi+DirEntry_FstClusLO] ; 26
15001 00010C2B A3[548E0100]      <1>      mov     [u.r0], eax ; First Cluster
15002      <1>
15003      <1>      ;mov     esi, FindFile_DirEntry
15004 00010C30 E8A2110000      <1>      call    get_file_name
15005      <1>
15006      <1>      ; 25/08/2024
15007      <1>      ; ecx <= 7 (from 'get_file_name')
15008      <1>      ;mov     cl, [FFF_Valid] ; (*)
15009 00010C35 BE[24880100]      <1>      mov     esi, DTA ; FFF data transfer address
15010      <1>
15011      <1> sysfff_6:
15012      <1>      ; 25/08/2024
15013      <1>      ;sub     ecx, ecx
15014 00010C3A 8A0D[58830100]      <1>      mov     cl, [FFF_Valid] ; (*) ecx <= 128
15015      <1>
15016 00010C40 8B3D[54830100]      <1>      mov     edi, [FFF_UBuffer] ; user's buffer address (edx)
15017 00010C46 E8C0FEFFFF      <1>      call    transfer_to_user_buffer
15018      <1>
15019 00010C4B 890D[548E0100]      <1>      mov     [u.r0], ecx ; actual transfer count
15020 00010C51 E844110000      <1>      call    reset_working_path
15021 00010C56 E923BEFFFF      <1>      jmp     sysret
15022      <1>
15023      <1> sysfff_7:
15024      <1>      ; 25/08/2024
15025 00010C5B C605[58830100]80      <1>      mov     byte [FFF_Valid], 128 ; (*)
15026      <1> sysfnf_11:
15027      <1>      ; ecx = 128
15028      <1>      ; 25/08/2024 (bugfix)
15029      <1>      ; 08/08/2022
15030 00010C62 BE[02800100]      <1>      mov     esi, FindFile_Drv
15031      <1>      ; 25/08/2024
15032 00010C67 29C9      <1>      sub     ecx, ecx ; 0
15033 00010C69 EBCF      <1>      jmp     short sysfff_6
15034      <1>
15035      <1> sysfnf: ; <Find Next File>
15036      <1>      ; 19/12/2025 (TRDOS 386 v2.0.10)
15037      <1>      ; 25/08/2024 (TRDOS 386 kernel v2.0.9)
15038      <1>      ; 29/08/2023 (TRDOS 386 kernel v2.0.6)
15039      <1>      ; 08/08/2022
15040      <1>      ; 30/07/2022 (TRDOS 386 kernel v2.0.5)
15041      <1>      ; 16/10/2016 TRDOS 386 (TRDOS v2.0) feature only !
15042      <1>      ; -derived from TRDOS v1.0, INT_21H.ASM-
15043      <1>      ; ("loc_INT21h_find_next_file")
15044      <1>      ; TRDOS 8086 (v1.0)
15045      <1>      ; 07/08/2011
15046      <1>      ; Find First File
15047      <1>      ; INPUT:
15048      <1>      ; none
15049      <1>      ; MSDOS OUTPUT:
15050      <1>      ; DTA: (Default address: PSP offset 80h)
15051      <1>      ; Offset Description
15052      <1>      ; 0 Reserved for use find next file
15053      <1>      ; 21 Attribute of file found
15054      <1>      ; 22 Time stamp of file
15055      <1>      ; 24 Date stamp of file
15056      <1>      ; 26 File size in bytes
15057      <1>      ; 30 Filename and extension (zero terminated)
15058      <1>      ; If cf = 1:
15059      <1>      ; Error Codes: (in AX)
15060      <1>      ; 18 - No more files
15061      <1>      ;
15062      <1>      ; TRDOS 386 (v2.0)
15063      <1>      ; 16/10/2016
15064      <1>      ;
15065      <1>      ; INPUT ->
15066      <1>      ; none
15067      <1>      ; OUTPUT ->
15068      <1>      ; EAX = 0 if CH input of 'Find First File' > 0
15069      <1>      ; EAX = First cluster number of file
15070      <1>      ; if CH input of 'Find First File' = 0
15071      <1>      ; EDX = File parameters table/structure address
15072      <1>      ;
15073      <1>      ; cf = 1 -> Error code in AL
15074      <1>      ;
15075      <1>      ; Modified Registers: EAX (at the return of system call)
15076      <1>
15077      <1>      ; Note: If byte [FFF_Valid] = 0
15078      <1>      ; 'sysfnf' will return with 'no more files' error.
15079      <1>      ; If byte [FFF_Valid] = 24
15080      <1>      ; 'sysfnf' will return with 24 bytes basic parameters
15081      <1>      ; at the address which is in EDX.
15082      <1>      ; If byte [FFF_Valid] = 128
15083      <1>      ; 'sysfnf' will return with 128 bytes Find File
15084      <1>      ; Structure/Table at the address which is in EDX.

```

```

15085 <1>
15086 00010C6B 803D[58830100]00 <1> cmp byte [FFF_Valid], 0
15087 00010C72 7713 <1> ja short stsfnf_0
15088 <1> ; 'no more files !' error
15089 <1> ;mov eax, ERR_NO_MORE_FILES ; 12
15090 <1> ; 30/07/2022
15091 00010C74 29C0 <1> sub eax, eax
15092 00010C76 B00C <1> mov al, ERR_NO_MORE_FILES ; 12
15093 00010C78 A3[548E0100] <1> mov [u.r0], eax
15094 00010C7D A3[C08E0100] <1> mov [u.error], eax
15095 00010C82 E9D7BDFFFF <1> jmp error
15096 <1>
15097 <1> stsfnf_0:
15098 <1> ;cmp byte [FFF_Valid], 128
15099 <1> ;je short stsfnf_1
15100 <1> ;cmp byte [FFF_Valid], 24
15101 <1> ;je short stsfnf_1
15102 <1> ;mov [FFF_Valid], 24 ; Default
15103 <1> stsfnf_1:
15104 00010C87 0FB61D[62770100] <1> movzx ebx, byte [Current_Drv]
15105 00010C8E 66891D[5E830100] <1> mov [SWP_DRV], bx
15106 00010C95 8A15[02800100] <1> mov dl, [FindFile_Drv]
15107 00010C9B 38DA <1> cmp dl, bl
15108 00010C9D 7532 <1> jne short stsfnf_2
15109 00010C9F 86DF <1> xchg bh, bl
15110 00010CA1 BE00010900 <1> mov esi, Logical_DOSDisks
15111 00010CA6 01DE <1> add esi, ebx
15112 00010CA8 EB34 <1> jmp short sysfnf_3
15113 <1>
15114 <1> sysfnf_8:
15115 00010CAA E886B4FFFF <1> call load_FAT_sub_directory
15116 00010CAF 7269 <1> jc short sysfnf_err_1 ; read error (no FNF stop)
15117 <1>
15118 <1> sysfnf_9:
15119 00010CB1 E8617DFFFF <1> call find_next_file
15120 00010CB6 7257 <1> jc short sysfnf_5
15121 <1> ; 25/08/2024
15122 <1> ; esi = Directory Entry (FindFile_DirEntry) Location
15123 <1> ; 08/08/2022
15124 <1> ; esi = FindFile_Drv ; wrong ! ; 25/08/2024
15125 <1>
15126 00010CB8 A0[59830100] <1> mov al, [FFF_Attrib]
15127 <1> ;or al, al
15128 <1> ;jz short sysfnf_10 ; 0 = No filter
15129 00010CBD 34FF <1> xor al, 0FFh
15130 00010CBF 20D8 <1> and al, bl
15131 00010CC1 75EE <1> jnz short sysfnf_9 ; search for next file until
15132 <1> ; an error return from
15133 <1> ; find_next_file procedure
15134 <1> sysfnf_10:
15135 <1> ;movzx ecx, byte [FFF_Valid]
15136 <1> ;cmp cl, 128 ; complete FindFile structure/table
15137 <1> ;je sysfnf_11
15138 <1> ;cmp cl, 24 ; basic parameters
15139 <1> ;je sysfnf_12
15140 <1> ;jmp sysfnf_12
15141 <1> ; 30/07/2022
15142 <1> ;movzx ecx, byte [FFF_Valid]
15143 <1> ; 25/08/2024
15144 <1> ;cmp cl, 128
15145 00010CC3 803D[58830100]80 <1> cmp byte [FFF_Valid], 128
15146 <1> ;jne short sysfnf_12
15147 <1> ;jmp short sysfnf_11 (*)
15148 <1> ; 08/08/2022
15149 <1> ;je short sysfnf_6 ; esi = FindFile_Drv
15150 <1> ; 29/08/2023 (BugFix)
15151 <1> ;je short sysfff_6 ; esi = FindFile_Drv
15152 <1> ; 25/08/2024 (BugFix of BugFix) (*)
15153 00010CCA 7496 <1> je short sysfnf_11 ; esi <> FindFile_Drv
15154 <1>
15155 00010CCC E937FFFFFF <1> jmp sysfnf_12
15156 <1>
15157 <1> stsfnf_2:
15158 00010CD1 FE05[5F830100] <1> inc byte [SWP_DRV_chg]
15159 <1>
15160 00010CD7 E8046AFFFF <1> call change_current_drive
15161 00010CDC 723C <1> jc short sysfnf_err_1 ; read error !
15162 <1> ; (do not stop, because
15163 <1> ; we don't have a
15164 <1> ; 'no more files'
15165 <1> ; -file not found- error,
15166 <1> ; next sysfnf system call
15167 <1> ; may solve the problem,
15168 <1> ; after re-placing the disk)
15169 <1> sysfnf_3:
15170 00010CDE A1[78800100] <1> mov eax, [FindFile_DirCluster]
15171 00010CE3 21C0 <1> and eax, eax
15172 00010CE5 7547 <1> jnz short sysfnf_6
15173 <1>
15174 00010CE7 803D[61770100]02 <1> cmp byte [Current_FATType], 2
15175 00010CEE 7723 <1> ja short sysfnf_err_0 ; invalid, we need to stop !?
15176 <1>
15177 <1> ; 19/12/2025
15178 <1> %if 0
15179 <1> cmp byte [Current_FATType], 1
15180 <1> jb short sysfnf_err_0 ; invalid, we need to stop !?
15181 <1> %endif
15182 <1>
15183 00010CF0 3805[897E0100] <1> cmp byte [DirBuff_validData], al ; 0
15184 00010CF6 7608 <1> jna short sysfnf_4
15185 <1>
15186 00010CF8 3805[8E7E0100] <1> cmp eax, [DirBuff_Cluster] ; 0 ?
15187 00010CFE 74B1 <1> je short sysfnf_9
15188 <1>
15189 <1> ;cmp byte [Current_Dir_Level], 0
15190 <1> ;ja short sysfnf_4
15191 <1> ;jna short sysfnf_9
15192 <1>
15193 <1> sysfnf_4:
15194 00010D00 FE05[5F830100] <1> inc byte [SWP_DRV_chg]
15195 00010D06 E8ACB3FFFF <1> call load_FAT_root_directory
15196 00010D0B 73A4 <1> jnc short sysfnf_9
15197 <1> ; eax = error code (17, 'drv not ready or read error')
15198 00010D0D EB0B <1> jmp short sysfnf_err_1 ; read error ! (no FNF stop)
15199 <1> ; (if you want, try again,
15200 <1> ; after re-placing the disk)
15201 <1> sysfnf_5:
15202 00010D0F 3C0C <1> cmp al, 12 ; 'no more files' error
15203 00010D11 7507 <1> jne short sysfnf_err_1 ; (no FNF stop -sysfnf will try
15204 <1> ; to read the directory again,
15205 <1> ; if the user calls sysfnf
15206 <1> ; just after this error return-)
15207 <1> ; (FNF stop -sysfnf will not try
15208 <1> ; to read the directory again-)

```

```

15209 <1>
15210 <1> sysfnf_err_0:
15211 00010D13 C605[58830100]00 <1> mov byte [FFF_Valid], 0 ; FNF stop sign
15212 <1> sysfnf_err_1:
15213 00010D1A A3[548E0100] <1> mov [u.r0], eax
15214 00010D1F A3[C08E0100] <1> mov [u.error], eax
15215 00010D24 E871100000 <1> call reset_working_path
15216 00010D29 E930BDFFFF <1> jmp error
15217 <1>
15218 <1> sysfnf_6:
15219 00010D2E 803D[897E0100]00 <1> cmp byte [DirBuff_validData], 0
15220 00010D35 760D <1> jna short sysfnf_7
15221 <1>
15222 00010D37 3B05[8E7E0100] <1> cmp eax, [DirBuff_Cluster]
15223 <1> ;je short sysfnf_9
15224 <1> ; 08/08/2022
15225 00010D3D 7505 <1> jne short sysfnf_7
15226 00010D3F E96DFFFFFF <1> jmp sysfnf_9
15227 <1> sysfnf_7:
15228 00010D44 FE05[5F830100] <1> inc byte [SWP_DRV_chg]
15229 <1>
15230 <1> ; 19/12/2025
15231 <1> %if 0
15232 <1> cmp byte [Current_FATtype], 1
15233 <1> ;jnb short sysfnf_8
15234 <1> ; 08/08/2022
15235 <1> ;jb short sysfnf_13
15236 <1> %endif
15237 00010D4A E95BFFFFFF <1> jmp sysfnf_8
15238 <1>
15239 <1> ; 19/12/2025
15240 <1> %if 0
15241 <1> sysfnf_13:
15242 <1> ; Singlix (TRFS) File System
15243 <1> ; (access via compatibility buffer)
15244 <1> call load_FS_sub_directory
15245 <1> ;jnc short sysfnf_9
15246 <1> ;jmp short sysfnf_err_1 ; read error (no FNF stop)
15247 <1> ; 08/08/2022
15248 <1> ;jc short sysfnf_err_1
15249 <1> jmp sysfnf_9
15250 <1> %endif
15251 <1>
15252 <1> writei:
15253 <1> ; 02/09/2024
15254 <1> ; 27/08/2024 - TRDOS 386 v2.0.9
15255 <1> ; 08/08/2022
15256 <1> ; 30/07/2022
15257 <1> ; 23/07/2022 - TRDOS 386 kernel v2.0.5
15258 <1> ; 26/10/2016
15259 <1> ; 25/10/2016
15260 <1> ; 23/10/2016
15261 <1> ; 22/10/2016
15262 <1> ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)
15263 <1> ; 19/05/2015 - 20/05/2015 (Retro UNIX 386 v1)
15264 <1> ; 12/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
15265 <1> ;
15266 <1> ; write data to file with first cluster number in EAX
15267 <1> ;
15268 <1> ; INPUTS ->
15269 <1> ; EAX - First cluster number of the file
15270 <1> ; EBX - File number (Open file index number)
15271 <1> ; u.count - byte count to be written
15272 <1> ; u.base - points to user buffer
15273 <1> ; u.fofp - points to dword with current file offset
15274 <1> ; i.size - file size
15275 <1> ; cdev - logical dos drive number of the file
15276 <1> ; OUTPUTS ->
15277 <1> ; u.count - cleared
15278 <1> ; u.nread - accumulates total bytes passed back
15279 <1> ; i.size - new file size (if file byte offset overs file size)
15280 <1> ; u.fofp - points to u.off (with new offset value)
15281 <1> ;
15282 <1> ; (Retro UNIX Prototype : 11/11/2012 - 18/11/2012, UNIXCOPY.ASM)
15283 <1> ; ((Modified registers: eax, edx, ebx, ecx, esi, edi, ebp))
15284 <1>
15285 00010D4F 31C9 <1> xor ecx, ecx
15286 00010D51 890D[808E0100] <1> mov [u.nread], ecx ; 0
15287 00010D57 66890D[B88E0100] <1> mov [u.pcount], cx ; 19/05/2015
15288 00010D5E 390D[7C8E0100] <1> cmp [u.count], ecx
15289 00010D64 7701 <1> ja short writei_1 ; 08/08/2022
15290 00010D66 C3 <1> retn
15291 <1> ; 23/07/2022
15292 <1> ;jna short dskw_8 ; retn
15293 <1>
15294 <1> writei_1:
15295 00010D67 C705[FC820100]FFFF- <1> mov dword [writei.fc lust], 0FFFFFFFh ; -1 ; reset
15296 00010D6F FFFF <1>
15297 <1> dskw:
15298 00010D71 881D[0C830100] <1> mov [writei.ofn], bl ; Open file number
15299 00010D77 880D[4F830100] <1> mov [setfmod], cl ; 0 ; reset 'update lm date&time' sign
15300 <1> dskw_0:
15301 <1> ; 26/10/2016
15302 <1> ; 22/10/2016, 23/10/2016, 25/10/2016
15303 <1> ; 19/10/2016 - TRDOS 386 (TRDOS v2.0)
15304 <1> ; 31/05/2015 - 25/07/2015 (Retro UNIX 386 v1)
15305 <1> ; 26/04/2013 - 20/09/2013 (Retro UNIX 8086 v1)
15306 <1> ;
15307 00010D7D E8D0000000 <1> ; 01/08/2013 (mkdir_w check)
15308 <1> call mget_w
15309 <1> ; eax = sector/block number
15310 00010D82 8B1D[688E0100] <1> mov ebx, [u.fofp]
15311 00010D88 8B13 <1> mov edx, [ebx]
15312 00010D8A 81E2FF010000 <1> and edx, 1FFh ; / test the lower 9 bits of the file offset
15313 00010D90 750C <1> jnz short dskw_1 ; / if its non-zero, branch
15314 <1> ; if zero, file offset = 0,
15315 <1> ; / 512, 1024,...(i.e., start of new block)
15316 00010D92 813D[7C8E0100]0002- <1> cmp dword [u.count], 512
15317 00010D9A 0000 <1>
15318 <1> ; / if zero, is there enough data to fill
15319 00010D9C 7331 <1> jnb short dskw_2 ; / an entire block? (i.e., no. of
15320 <1> ; bytes to be written greater than 512.?
15321 <1> ; Yes, branch. Don't have to read block
15322 <1> dskw_1: ; in as no past info. is to be saved
15323 <1> ; (the entire block will be overwritten).
15324 <1> ; 23/10/2016
15325 00010D9E BB[84920100] <1> mov ebx, writei_buffer
15326 <1> ; esi = logical dos drive description table address
15327 <1> ; eax = sector number
15328 <1> ; ebx = buffer address (in kernel's memory space)
15329 <1> ; ecx = sector count
15330 <1> ; 30/07/2022

```

```

15331      <1>      ;mov     ecx, 1
15332 00010DA3 31C9      <1>      xor     ecx, ecx
15333 00010DA5 FEC1      <1>      inc     cl
15334      <1>      ; ecx = 1
15335 00010DA7 E8370C0000 <1>      call    disk_read
15336      <1>      ;call    dskrd ; / no, must retain old info..
15337      <1>      ; / Hence, read block 'r1' into an I/O buffer
15338 00010DAC 7321      <1>      jnc     short dskw_2
15339      <1>
15340      <1>      ; disk read error
15341      <1>      ; 30/07/2022
15342      <1>      ;mov     eax, 17 ; drive not ready or READ ERROR !
15343 00010DAE 29C0      <1>      sub     eax, eax
15344 00010DB0 B011      <1>      mov     al, 17
15345      <1>      dskw_err: ; jump from disk write error
15346 00010DB2 A3[548E0100] <1>      mov     [u.r0], eax
15347 00010DB7 A3[C08E0100] <1>      mov     [u.error], eax
15348      <1>
15349 00010DBC 803D[4F830100]00 <1>      cmp     byte [setfmod], 0
15350      <1>      ;jna     error
15351      <1>      ; 23/07/2022
15352 00010DC3 7605      <1>      jna     short writei_err
15353      <1>
15354 00010DC5 E887020000 <1>      call    update_file_lmdt ; update last modif. date&time of the file
15355      <1>      ;mov     byte [setfmod], 0
15356      <1>      writei_err:
15357 00010DCA E98FBCFFFF <1>      jmp     error
15358      <1>
15359      <1>      dskw_2: ; 3:
15360      <1>      ; 23/10/2016
15361 00010DCF C605[E8820100]01 <1>      mov     byte [writei.valid], 1 ; writei buffer contains valid data
15362 00010DD6 56      <1>      push    esi ; logical dos drive description table address
15363      <1>      ; EAX (r1) = block/sector number
15364      <1>      ;call    wslot
15365      <1>      ; jsr r0,wslot / set write and inhibit bits in I/O queue,
15366      <1>      ; / proc. status=0, r5 points to 1st word of data
15367 00010DD7 803D[BA8E0100]00 <1>      cmp     byte [u.kcall], 0
15368 00010DDE 770F      <1>      ja      short dskw_4 ; zf=0 -> the caller is 'mkdir'
15369      <1>
15370 00010DE0 66833D[B88E0100]00 <1>      cmp     word [u.pcount], 0
15371 00010DE8 7705      <1>      ja      short dskw_4
15372      <1>      dskw_3:
15373      <1>      ; [u.base] = virtual address to transfer (as source address)
15374 00010DEA E844FAFFFF <1>      call    trans_addr_r ; translate virtual address to physical (r)
15375      <1>      dskw_4:
15376 00010DEF BB[84920100] <1>      mov     ebx, writei_buffer
15377      <1>      ; EBX (r5) = system (I/O) buffer address
15378 00010DF4 E8A6FAFFFF <1>      call    storg
15379      <1>      ; ESI = file (user data) offset
15380      <1>      ; EDI = sector (I/O) buffer offset
15381      <1>      ; ECX = byte count
15382      <1>
15383 00010DF9 F3A4      <1>      rep     movsb
15384      <1>      ; 25/07/2015
15385      <1>      ; eax = remain bytes in buffer
15386      <1>      ; (check if remain bytes in the buffer > [u.pcount])
15387 00010DFB 09C0      <1>      or      eax, eax
15388 00010DFD 75EB      <1>      jnz     short dskw_3 ; (page end before system buffer end!)
15389      <1>
15390      <1>      ; 23/10/2016
15391 00010DFF B101      <1>      mov     cl, 1
15392 00010E01 5E      <1>      pop     esi
15393 00010E02 A1[EC820100] <1>      mov     eax, [writei.sector]
15394      <1>      ; esi = logical dos drive description table address
15395      <1>      ; eax = sector number
15396      <1>      ; ebx = writei buffer address
15397      <1>      ; ecx = sector count
15398 00010E07 E8C80B0000 <1>      call    disk_write ; / yes, write the block
15399 00010E0C 7313      <1>      jnc     short dskw_5
15400      <1>
15401      <1>      ;mov     eax, 18 ; drive not ready or WRITE ERROR !
15402      <1>      ; 30/08/2022
15403 00010E0E 29C0      <1>      sub     eax, eax
15404 00010E10 B012      <1>      mov     al, 18
15405 00010E12 EB9E      <1>      jmp     short dskw_err
15406      <1>
15407      <1>      dskw_7:
15408      <1>      ; update last modif. date&time of the file
15409      <1>      ; (also updates file size as OF_SIZE)
15410 00010E14 E838020000 <1>      call    update_file_lmdt
15411      <1>      ;mov     byte [setfmod], 0
15412      <1>
15413      <1>      ; 03/08/2013
15414 00010E19 C605[BA8E0100]00 <1>      mov     byte [u.kcall], 0
15415      <1>      ; 23/10/2016
15416      <1>      ;mov     eax, [writei.fclust]
15417      <1>      dskw_8: ; 23/07/2022
15418 00010E20 C3      <1>      retn
15419      <1>
15420      <1>      dskw_5:
15421      <1>      ; 26/10/2016
15422 00010E21 0FB61D[0C830100] <1>      movzx    ebx, byte [writei.ofn] ; open file number
15423 00010E28 C0E302      <1>      shl     bl, 2 ; *4
15424 00010E2B 8B83[64840100] <1>      mov     eax, [ebx+OF_POINTER]
15425 00010E31 3B83[E4840100] <1>      cmp     eax, [ebx+OF_SIZE]
15426 00010E37 7606      <1>      jna     short dskw_6
15427 00010E39 8983[E4840100] <1>      mov     [ebx+OF_SIZE], eax
15428      <1>      dskw_6:
15429      <1>      ;shr     bl, 2
15430 00010E3F 833D[7C8E0100]00 <1>      cmp     dword [u.count], 0 ; / any more data to write?
15431 00010E46 76CC      <1>      jna     short dskw_7
15432 00010E48 A1[FC820100] <1>      mov     eax, [writei.fclust]
15433 00010E4D E92BFFFFFF <1>      jmp     dskw_0 ; / yes, branch
15434      <1>
15435      <1>      mget_w:
15436      <1>      ; 19/12/2025 - TRDOS 386 v2.0.10
15437      <1>      ; 03/09/2024
15438      <1>      ; 02/09/2024
15439      <1>      ; 25/08/2024 - TRDOS 386 v2.0.9
15440      <1>      ; 08/08/2022
15441      <1>      ; 25/07/2022
15442      <1>      ; 23/07/2022 - TRDOS 386 kernel v2.0.5
15443      <1>      ; 02/11/2016
15444      <1>      ; 01/11/2016
15445      <1>      ; 23/10/2016, 31/10/2016
15446      <1>      ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)
15447      <1>      ; 03/06/2015 (Retro UNIX 386 v1, 'mget', u.5s)
15448      <1>      ; 22/03/2013 - 31/07/2013 (Retro UNIX 8086 v1)
15449      <1>
15450      <1>      ; Get existing or (allocate) a new disk block for file
15451      <1>
15452      <1>      ; INPUTS ->
15453      <1>      ; [u.fofp] = file offset pointer
15454      <1>      ; [i.size] = file size

```

```

15455 <1> ; [u.count] = byte count
15456 <1> ; EAX = First cluster
15457 <1> ; [cdev] = Logical dos drive number
15458 <1> ; [writei.ofn] = File Number
15459 <1> ; (Open file index, 0 based)
15460 <1> ; ([u.off] = file offset)
15461 <1> ; OUTPUTS ->
15462 <1> ; EAX = logical sector number
15463 <1> ; ESI = Logical Dos Drive Description Table address
15464 <1> ;
15465 <1> ; Modified registers: EDX, EBX, ECX, ESI, EDI, EBP
15466 <1>
15467 00010E52 8B35[688E0100] <1> mov esi, [u.fofp]
15468 00010E58 8B2E <1> mov ebp, [esi] ; u.off (or EBX*4+OF_POINTER)
15469 <1>
15470 00010E5A 29C9 <1> sub ecx, ecx
15471 00010E5C 8A2D[398E0100] <1> mov ch, [cdev]
15472 <1>
15473 00010E62 BE00010900 <1> mov esi, Logical_DOSDisks
15474 00010E67 01CE <1> add esi, ecx
15475 <1>
15476 <1> ; 31/10/2016
15477 00010E69 89C3 <1> mov ebx, eax ; First Cluster or FDT address
15478 <1>
15479 <1> ; 19/12/2025
15480 <1> %if 0
15481 <1> cmp byte [esi+LD_FATType], 0
15482 <1> ; jna mget_w_14 ; Singlix FS
15483 <1> ; 23/07/2022
15484 <1> ja short mget_w_20
15485 <1> jmp mget_w_14 ; Singlix FS
15486 <1> %endif
15487 <1>
15488 <1> mget_w_20:
15489 <1> ; movzx eax, word [esi+LD_BPB+BytesPerSec]
15490 <1> ; 19/12/2025
15491 00010E6B B800020000 <1> mov eax, 512
15492 00010E70 0FB65613 <1> movzx edx, byte [esi+LD_BPB+SecPerClust]
15493 00010E74 8815[EA820100] <1> mov [writei.spc], dl ; sectors per cluster
15494 00010E7A F7E2 <1> mul edx
15495 <1> ; edx = 0
15496 <1> ; eax = bytes per cluster (<= 65536)
15497 <1>
15498 <1> ; 02/11/2016
15499 00010E7C 89C1 <1> mov ecx, eax
15500 00010E7E 48 <1> dec eax
15501 00010E7F 66A3[F0820100] <1> mov [writei.bpc], ax
15502 <1>
15503 00010E85 89E8 <1> mov eax, ebp
15504 00010E87 0305[7C8E0100] <1> add eax, [u.count] ; next file position
15505 00010E8D 3B05[608F0100] <1> cmp eax, [i.size] ; <= file size ?
15506 <1> ; jna mget_w_4 ; no
15507 <1> ; 23/07/2022
15508 00010E93 7705 <1> ja short mget_w_21
15509 <1> ; 02/09/2024
15510 <1> ; ebx = first cluster (input)
15511 00010E95 E925010000 <1> jmp mget_w_4
15512 <1> mget_w_21:
15513 <1> div ecx
15514 00010E9C A3[F8820100] <1> mov [writei.c_index], eax ; cluster index
15515 <1> ; edx = byte offset in cluster (<= 65535)
15516 <1> ; mov [writei.offset], dx
15517 <1> ; shr dx, 9 ; / 512
15518 <1> ; mov [writei.s_index], dl ; sector index in cluster (0 to spc -1)
15519 <1>
15520 00010EA1 29D2 <1> sub edx, edx ; 01/11/2016
15521 00010EA3 8915[EC820100] <1> mov [writei.sector], edx ; 0
15522 00010EA9 668915[F2820100] <1> mov [writei.offset], dx ; byte offset in cluster
15523 00010EB0 8815[EB820100] <1> mov [writei.s_index], dl ; sector index in cluster (0 to spc -1)
15524 <1>
15525 00010EB6 89D8 <1> mov eax, ebx ; First Cluster
15526 <1>
15527 <1> ; is this the 1st mget_w or a next mget_w call ? (by 'writei')
15528 00010EB8 3815[E8820100] <1> cmp byte [writei.valid], dl ; 0
15529 00010EBE 7624 <1> jna short mget_w_0
15530 <1>
15531 00010EC0 8815[E8820100] <1> mov byte [writei.valid], dl ; 0 ; reset ('writei' will set it)
15532 <1>
15533 00010EC6 3B05[FC820100] <1> cmp eax, [writei.fclust]
15534 00010ECC 7516 <1> jne short mget_w_0
15535 <1>
15536 00010ECE 8A0D[398E0100] <1> mov cl, [cdev]
15537 00010ED4 3A0D[E9820100] <1> cmp cl, [writei.driv]
15538 00010EDA 7508 <1> jne short mget_w_0
15539 <1> ; [writei.l_clust] & [writei.l_index] are valid,
15540 <1> ; we don't need to get last cluster & last cluster index
15541 00010EDC 8B0D[08830100] <1> mov ecx, [writei.l_index]
15542 00010EE2 EB7F <1> jmp short mget_w_2
15543 <1> mget_w_0:
15544 00010EE4 A3[FC820100] <1> mov [writei.fclust], eax ; first cluster
15545 <1> ; edx = 0
15546 00010EE9 A3[F4820100] <1> mov [writei.cluster], eax ; first cluster ; 01/11/2016
15547 00010EEE 8915[00830100] <1> mov [writei.fs_index], edx ; 0 ; current cluster index
15548 <1>
15549 <1> ;;;
15550 <1> ; 25/08/2024 - TRDOS 386 v2.0.9
15551 00010EF4 09C0 <1> or eax, eax ; is first cluster number = 0 ?
15552 00010EF6 751A <1> jnz short mget_w_27 ; no
15553 <1>
15554 <1> ; eax = 0 (*)
15555 <1> ; ((a directory entry with zero file size)) (*)
15556 <1> ; (*) (((('syswrite' may be used for writing to an empty file.
15557 <1> ; Before this modification, 'syscreat' and 'syswrite' system calls
15558 <1> ; did wrong things together... 'writetest.s' file, 24/08/2024.)))
15559 <1>
15560 00010EF8 E8FFB8FFFF <1> call add_new_cluster ; (*) 2024 (v2.0.9) modification
15561 00010EFD 721A <1> jc short mget_w_errj ; eax = error code
15562 <1> ; eax = (new) first cluster
15563 <1> ; skip 'get_last_cluster' and set ecx to -1
15564 <1> ;
15565 <1> ; 02/09/2024
15566 00010EFF A3[FC820100] <1> mov [writei.fclust], eax ; first cluster
15567 00010F04 A3[F4820100] <1> mov [writei.cluster], eax
15568 <1> ; 03/09/2024
15569 00010F09 A3[50830100] <1> mov [setfclust], eax
15570 <1> ;
15571 00010F0E 31C9 <1> xor ecx, ecx
15572 00010F10 EB12 <1> jmp short mget_w_28
15573 <1> ;;;
15574 <1>
15575 <1> mget_w_27:
15576 <1> ; FAT file system (FAT12, FAT16, FAT32)
15577 00010F12 E8C8B7FFFF <1> call get_last_cluster
15578 <1> ; jc short mget_w_err ; eax = error code

```

```

15579      ; 08/08/2022
15580 00010F17 7305      <1>      jnc      short mget_w_25
15581      <1> mget_w_errj:      ; 25/08/2024
15582 00010F19 E992000000      <1>      jmp      mget_w_err
15583      <1>
15584      <1> mget_w_25:
15585      <1>      ; 25/08/2024
15586 00010F1E 8B0D[30810100]      <1>      mov     ecx, [glc_index] ; last cluster index
15587      <1> mget_w_28:      ; 02/09/2024
15588 00010F24 890D[08830100]      <1>      mov     [writei.l_index], ecx
15589 00010F2A A3[04830100]      <1>      mov     [writei.lclust], eax ; last cluster
15590      <1>
15591 00010F2F A0[0C830100]      <1>      mov     al, [writei.ofn]
15592 00010F34 FEC0      <1>      inc     al
15593 00010F36 A2[4F830100]      <1>      mov     [setfmod], al ; update lm date&time sign
15594      <1>
15595      <1> mget_w_1:
15596 00010F3B 3B0D[F8820100]      <1>      cmp     ecx, [writei.c_index] ; last cluster index
15597 00010F41 7320      <1>      jnb     short mget_w_2 ; 01/11/2016
15598      <1>
15599 00010F43 A1[04830100]      <1>      mov     eax, [writei.lclust]
15600      <1>      ; EAX = Last cluster
15601 00010F48 E8AFB8FFFF      <1>      call    add_new_cluster
15602 00010F4D 7261      <1>      jc      short mget_w_err ; eax = error code
15603      <1>      ; edx = 0
15604 00010F4F A3[04830100]      <1>      mov     [writei.lclust], eax ; (new) last cluster
15605 00010F54 8B0D[08830100]      <1>      mov     ecx, [writei.l_index]
15606 00010F5A 41      <1>      inc     ecx ; add 1 to last cluster index
15607 00010F5B 890D[08830100]      <1>      mov     [writei.l_index], ecx ; current last cluster index
15608      <1>
15609 00010F61 EBD8      <1>      jmp     short mget_w_1
15610      <1>
15611      <1> mget_w_2:
15612 00010F63 89E9      <1>      mov     ecx, ebp
15613 00010F65 030D[7C8E0100]      <1>      add     ecx, [u.count]
15614 00010F6B 890D[608F0100]      <1>      mov     [i.size], ecx ; save new file size
15615      <1>      ;sub     edx, edx ; 0
15616      <1>
15617 00010F71 A0[398E0100]      <1>      mov     al, [cdev]
15618 00010F76 A2[E9820100]      <1>      mov     [writei.driv], al ; physical drive number
15619      <1>      ; edx = 0
15620 00010F7B 89E8      <1>      mov     eax, ebp ; file offset
15621 00010F7D 0FB70D[F0820100]      <1>      movzx   ecx, word [writei.bpc] ; bytes per cluster - 1
15622 00010F84 41      <1>      inc     ecx ; bytes per cluster
15623 00010F85 F7F1      <1>      div     ecx
15624      <1>      ; edx = byte offset in cluster (<= 65535)
15625      <1>      ; eax = cluster index
15626 00010F87 A3[F8820100]      <1>      mov     [writei.c_index], eax
15627 00010F8C 668915[F2820100]      <1>      mov     [writei.offset], dx
15628      <1>      ;shr     dx, 9 ; / 512
15629      <1>      ; 23/07/2022
15630 00010F93 C1EA09      <1>      shr     edx, 9
15631 00010F96 8815[EB820100]      <1>      mov     [writei.s_index], dl ; sector index in cluster (0 to spc -1)
15632      <1>
15633      <1> mget_w_3:
15634 00010F9C 3B05[08830100]      <1>      cmp     eax, [writei.l_index] ; last cluster index
15635 00010FA2 753E      <1>      jne     short mget_w_5
15636      <1>
15637 00010FA4 A3[00830100]      <1>      mov     [writei.fs_index], eax ; cluster index (for next check)
15638      <1>
15639 00010FA9 A1[04830100]      <1>      mov     eax, [writei.lclust] ; last cluster
15640 00010FAE EB74      <1>      jmp     short mget_w_10
15641      <1>
15642      <1> mget_w_err:
15643 00010FB0 A3[C08E0100]      <1>      mov     [u.error], eax
15644 00010FB5 A3[548E0100]      <1>      mov     [u.r0], eax
15645 00010FBA E99FBAFFFF      <1>      jmp     error
15646      <1>
15647      <1> mget_w_4: ; 02/11/2016
15648      <1>      ; eax = next file position
15649 00010FBF 2B05[7C8E0100]      <1>      sub     eax, [u.count] ; current file position
15650      <1>      ; edx = 0
15651      <1>      ; ecx = bytes per cluster
15652 00010FC5 F7F1      <1>      div     ecx
15653 00010FC7 A3[F8820100]      <1>      mov     [writei.c_index], eax ; cluster index
15654 00010FCC 668915[F2820100]      <1>      mov     [writei.offset], dx
15655      <1>      ;shr     dx, 9 ; / 512
15656      <1>      ; 23/07/2022
15657 00010FD3 C1EA09      <1>      shr     edx, 9
15658 00010FD6 8815[EB820100]      <1>      mov     [writei.s_index], dl ; sector index in cluster (0 to spc -1)
15659      <1>
15660      <1>      ; 02/09/2024
15661 00010FDC 891D[FC820100]      <1>      mov     [writei.fclust], ebx
15662      <1>
15663      <1> mget_w_5:
15664 00010FE2 21C0      <1>      and     eax, eax ; 0 = First Cluster's index number
15665 00010FE4 750C      <1>      jnz     short mget_w_6
15666      <1>
15667 00010FE6 A3[00830100]      <1>      mov     [writei.fs_index], eax ; cluster index (for next check)
15668      <1>
15669 00010FEB A1[FC820100]      <1>      mov     eax, [writei.fclust] ; first cluster
15670 00010FF0 EB32      <1>      jmp     short mget_w_10
15671      <1>
15672      <1> mget_w_6:
15673 00010FF2 3B05[00830100]      <1>      cmp     eax, [writei.fs_index] ; current cluster index (>0)
15674 00010FF8 7507      <1>      jne     short mget_w_7
15675 00010FFA A1[F4820100]      <1>      mov     eax, [writei.cluster] ; current cluster
15676 00010FFF EB34      <1>      jmp     short mget_w_11
15677      <1>
15678      <1> mget_w_7:
15679 00011001 89C1      <1>      mov     ecx, eax
15680 00011003 2B0D[00830100]      <1>      sub     ecx, [writei.fs_index]
15681 00011009 730D      <1>      jnc     short mget_w_8
15682      <1>      ; get cluster by index from the first cluster
15683 0001100B A1[FC820100]      <1>      mov     eax, [writei.fclust]
15684 00011010 8B0D[F8820100]      <1>      mov     ecx, [writei.c_index]
15685 00011016 EB05      <1>      jmp     short mget_w_9
15686      <1>
15687      <1> mget_w_8:
15688 00011018 A1[F4820100]      <1>      mov     eax, [writei.cluster] ; beginning cluster
15689      <1>      ; ecx = cluster sequence number after the beginning cluster
15690      <1>      ; sub     edx, edx ; 0
15691      <1>
15692      <1> mget_w_9:
15693      <1>      ; EAX = Beginning cluster
15694      <1>      ; EDX = Sector index in disk/file section
15695      <1>      ; (Only for SINGLIX file system!)
15696      <1>      ; ECX = Cluster sequence number after the beginning cluster
15697      <1>      ; ESI = Logical DOS Drive Description Table address
15698 0001101D E887B8FFFF      <1>      call    get_cluster_by_index
15699 00011022 728C      <1>      jc      short mget_w_err ; error code in EAX
15700      <1>
15701      <1>      ; EAX = Cluster number
15702      <1> mget_w_10:

```

```

15703 00011024 A3[F4820100] <1> mov [writei.cluster], eax ; FDT number for Singlix File System
15704 <1>
15705 <1> ; 19/12/2025
15706 <1> %if 0
15707 <1> cmp byte [esi+LD_FATType], 0
15708 <1> jna short mget_w_13
15709 <1> %endif
15710 <1> ; 01/11/2016
15711 00011029 8B15[F8820100] <1> mov edx, [writei.c_index]
15712 0001102F 8915[00830100] <1> mov [writei.fs_index], edx
15713 <1> mget_w_11:
15714 <1> ;sub eax, 2
15715 <1> ; 23/07/2022
15716 00011035 48 <1> dec eax
15717 00011036 48 <1> dec eax
15718 00011037 0FB615[EA820100] <1> movzx edx, byte [writei.spc]
15719 0001103E F7E2 <1> mul edx
15720 <1>
15721 00011040 034668 <1> add eax, [esi+LD_DATABegin]
15722 00011043 8A15[EB820100] <1> mov dl, [writei.s_index]
15723 00011049 01D0 <1> add eax, edx
15724 <1> mget_w_12:
15725 0001104B A3[EC820100] <1> mov [writei.sector], eax
15726 <1> ; buffer validation must be done in writei
15727 <1> ; mov byte [writei.valid], 1
15728 00011050 C3 <1> ret
15729 <1>
15730 <1> ; 19/12/2025
15731 <1> %if 0
15732 <1>
15733 <1> mget_w_13:
15734 <1> ; EAX = FDT number (Current Section)
15735 <1> ; EDX = Sector index from the first section (0,1,2,3,4...)
15736 <1> sub edx, [writei.fs_index]
15737 <1> ; EDX = Sector index from current section
15738 <1> mov [writei.fs_index], edx
15739 <1> inc eax ; the first data sector in FS disk section
15740 <1> add eax, edx
15741 <1> jmp short mget_w_12
15742 <1>
15743 <1> mget_w_14:
15744 <1> ;mov cl, [esi+LD_FS_BytesPerSec+1]
15745 <1> ;shr cl, 1 ; 1 for 512 bytes, 4 for 2048 bytes
15746 <1> ;mov [writei.spc], cl ; sectors per cluster
15747 <1> ; 19/12/2025
15748 <1> mov byte [writei.spc], 1
15749 <1> ; NOTE: writei bytes per sector value is always 512 !
15750 <1> mov word [writei.bpc], 512
15751 <1>
15752 <1> mov ecx, ebp
15753 <1> add ecx, [u.count] ; next file position
15754 <1> cmp ecx, [i.size] ; <= file size ?
15755 <1> ;jna mget_w_19 ; no
15756 <1> ja short mget_w_22
15757 <1> ; 23/07/2022
15758 <1> jmp mget_w_19
15759 <1>
15760 <1> mget_w_22:
15761 <1> sub edx, edx ; 0
15762 <1> mov [writei.sector], edx ; 0
15763 <1> mov [writei.offset], dx ; byte offset in cluster
15764 <1> mov [writei.s_index], dl ; sector index in cluster (0 to spc -1)
15765 <1>
15766 <1> shr ecx, 9 ; 1 cluster = 512 bytes
15767 <1> mov [writei.c_index], ecx ; section/cluster index
15768 <1>
15769 <1> mov eax, ebx ; FDT number (First FDT address)
15770 <1>
15771 <1> ; is this the 1st mget_w or a next mget_w call ? (by 'writei')
15772 <1> cmp byte [writei.valid], dl ; 0
15773 <1> jna short mget_w_15
15774 <1>
15775 <1> mov byte [writei.valid], dl ; 0 ; reset ('writei' will set it)
15776 <1>
15777 <1> cmp eax, [writei.fclust]
15778 <1> jne short mget_w_15
15779 <1>
15780 <1> mov cl, [cdev]
15781 <1> cmp cl, [writei.driv]
15782 <1> jne short mget_w_15
15783 <1> ; [writei.l_clust] & [writei.l_index] are valid,
15784 <1> ; we don't need to get last cluster & last cluster index
15785 <1> mov ecx, [writei.l_index]
15786 <1> jmp short mget_w_17
15787 <1> mget_w_15:
15788 <1> mov [writei.fclust], eax ; first section (FDT number)
15789 <1> ; edx = 0
15790 <1> mov [writei.cluster], edx ; 0 ; current section
15791 <1> mov [writei.fs_index], edx ; 0 ; current section index
15792 <1>
15793 <1> ; eax = FDT number (section 0 header address)
15794 <1> call get_last_section
15795 <1> ;jc short mget_w_err ; eax = error code
15796 <1> ; 08/08/2022
15797 <1> jnc short mget_w_26
15798 <1> jmp mget_w_err
15799 <1> mget_w_26:
15800 <1> mov [writei.fs_index], edx ; sector index in last section
15801 <1>
15802 <1> mov [writei.l_clust], eax ; last section address
15803 <1>
15804 <1> mov ecx, [glc_index] ; last section index
15805 <1> mov [writei.l_index], ecx
15806 <1>
15807 <1> mov al, [writei.ofn]
15808 <1> inc al
15809 <1> mov [setfmod], al ; update lm date&time sign
15810 <1>
15811 <1> mget_w_16:
15812 <1> ; edx = (existing) last section (sector) index
15813 <1> mov ecx, [writei.c_index] ; final section (sector) index
15814 <1> sub ecx, edx
15815 <1> jna short mget_w_19
15816 <1> ; ecx = sector count
15817 <1> mget_w_17:
15818 <1> mov eax, [writei.l_clust]
15819 <1> ; ESI = Logical dos drv desc. table address
15820 <1> ; EAX = Last section
15821 <1> ; (ECX = 0 for directory)
15822 <1> ; ECX = sector count (except FDT)
15823 <1> call add_new_fs_section
15824 <1> jnc short mget_w_18
15825 <1>
15826 <1> ; If error number = 27h (insufficient disk space)

```

```

15827 <1> ; it is needed to check free consequent sectors
15828 <1> ; (1 data sector at least and +1 section header sector)
15829 <1>
15830 <1> cmp     eax, 27h
15831 <1> ;jne    mget_w_err ; eax = error code
15832 <1> ; 25/07/2022
15833 <1> je      short mget_w_24
15834 <1> mget_w_23:
15835 <1> jmp     mget_w_err
15836 <1> mget_w_24:
15837 <1> ; ecx = count of free consequent sectors
15838 <1> ; ecx must be > 1 (1 data + 1 header sector)
15839 <1> dec     ecx
15840 <1> ;jz     short mget_w_err
15841 <1> ;jmp     short mget_w_17
15842 <1> ; 23/07/2022
15843 <1> jnz     short mget_w_17
15844 <1> ;jmp     mget_w_err
15845 <1> ; 25/07/2022
15846 <1> jmp     short mget_w_23
15847 <1>
15848 <1> mget_w_18:
15849 <1> mov     [writei.lclust], eax ; (new) last section
15850 <1> ; ecx = sector count (except section header)
15851 <1> mov     edx, [writei.l_index]
15852 <1> add     edx, ecx ; add sector count to index
15853 <1> mov     [writei.l_index], edx
15854 <1> jmp     short mget_w_16
15855 <1>
15856 <1> mget_w_19:
15857 <1> mov     ecx, ebp
15858 <1> add     ecx, [u.count]
15859 <1> mov     [i.size], ecx ; save new file size
15860 <1> ;sub     edx, edx ; 0
15861 <1>
15862 <1> mov     al, [cdev]
15863 <1> mov     [writei.driv], al ; physical drive number
15864 <1> ; edx = 0
15865 <1> mov     eax, ebp ; file offset
15866 <1> mov     edx, eax
15867 <1> ; 1 cluster = 512 bytes (for Singlix FS)
15868 <1> shr     eax, 9 ; / 512
15869 <1> and     edx, 1FFh
15870 <1> ; edx = byte offset in cluster/sector (<= 511)
15871 <1> ; eax = section (sector/cluster) index
15872 <1> mov     [writei.c_index], eax
15873 <1> mov     [writei.offset], dx
15874 <1> ;mov     byte [writei.s_index], 0 ; sector index in cluster
15875 <1> jmp     mget_w_3
15876 <1>
15877 <1> %endif
15878 <1>
15879 <1> update_file_lmdt: ; & update file size
15880 <1> ; 19/12/2025 - TRDOS 386 v2.0.10
15881 <1> ; 03/09/2024
15882 <1> ; 27/08/2024 - TRDOS 386 v2.0.9
15883 <1> ; 30/07/2022
15884 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
15885 <1> ; 26/10/2016
15886 <1> ; 24/10/2016
15887 <1> ; 23/10/2016
15888 <1> ; 22/10/2016 - TRDOS 386 (TRDOS v2.0)
15889 <1> ;
15890 <1> ; Update last modification date&time of file
15891 <1> ; (call from syswrite -> writei)
15892 <1> ; ((also updates file size)) // 26/10/2016
15893 <1> ;
15894 <1> ; INPUT:
15895 <1> ; byte [setfmod] = open file number
15896 <1> ; OUTPUT:
15897 <1> ; cf = 0 -> success !
15898 <1> ; cf = 1 -> lmdt update has been failed!
15899 <1> ;
15900 <1> ; Modified registers: eax, ebx, ecx, edx, esi, edi
15901 <1> ;
15902 <1>
15903 <1> ;cmp     byte [setfmod], 0
15904 <1> ;jna     short uflmdt_2 ; nothing to do
15905 <1>
15906 <1> ; 27/08/2024
15907 <1> ; if [ebx+OF_FCLUSTER] = 0 and [setfclust] > 0
15908 <1> ; [setfclust] will be copied to Directory entry's
15909 <1> ; first cluster fields (LO/HI).
15910 <1> ; (and [setfclust] will be cleared.)
15911 <1>
15912 <1> xor     eax, eax
15913 <1>
15914 <1> movzx   ebx, byte [setfmod]
15915 <1> dec     bl ; open file index number (0 based)
15916 <1>
15917 <1> mov     ah, [ebx+OF_DRIVE]
15918 <1> mov     esi, Logical_DOSDisks
15919 <1> add     esi, eax
15920 <1> shl     bl, 2 ; *4
15921 <1> mov     ecx, [ebx+OF_FCLUSTER] ; first cluster
15922 <1> mov     edx, [ebx+OF_DIRCLUSTER] ; dir cluster
15923 <1>
15924 <1> shr     bl, 1 ; /2
15925 <1> movzx   edi, word [ebx+OF_DIRENTRY]
15926 <1>
15927 <1> ; 03/09/2024
15928 <1> add     ah, 'A'
15929 <1> mov     al, [esi+LD_FATType]
15930 <1>
15931 <1> cmp     byte [DirBuff_ValidData], 1
15932 <1> jnb     short uflmdt_4
15933 <1>
15934 <1> ;mov     al, [DirBuff_DRV]
15935 <1> ;sub     al, 'A'
15936 <1> ;cmp     al, ah
15937 <1> ; 03/09/2024
15938 <1> ; 27/08/2024
15939 <1> ;add     ah, 'A'
15940 <1> ;mov     al, [esi+LD_FATType]
15941 <1> cmp     ah, [DirBuff_DRV]
15942 <1> jne     short uflmdt_4 ; different drive
15943 <1> ;mov     al, [esi+LD_FATType]
15944 <1> cmp     al, [DirBuff_FATType]
15945 <1> jne     short uflmdt_5 ; different FS type
15946 <1> cmp     edx, [DirBuff_Cluster]
15947 <1> jne     short uflmdt_5 ; different cluster
15948 <1>
15949 <1> uflmdt_1:
15950 <1> ; Directory buffer is ready here!

```



```

15951      <1>      ; OF_FCLUSTER must be compared/verified
15952 000110A8 BE00000800 <1>      mov     esi, Directory_Buffer
15953      <1>      ;shl     di, 5 ; dir entry index * 32
15954      <1>      ; 23/07/2022
15955 000110AD C1E705 <1>      shl     edi, 5
15956 000110B0 01FE <1>      add     esi, edi ; offset
15957      <1>      ;
15958      <1>      ; 03/09/2024
15959      <1>      ;test    byte [esi+DirEntry_Attr], 18h ; Vol & Dir
15960      <1>      ;jnz     short uflmdt_2 ; not a valid file !
15961 000110B2 668B4614 <1>      mov     ax, [esi+DirEntry_FstClusHI]
15962 000110B6 C1E010 <1>      shl     eax, 16
15963 000110B9 668B461A <1>      mov     ax, [esi+DirEntry_FstClusLO]
15964      <1>      ;
15965      <1>      ; 03/09/2024 (short jump)
15966 000110BD F6460B18 <1>      test    byte [esi+DirEntry_Attr], 18h ; Vol & Dir
15967 000110C1 7571 <1>      jnz     short uflmdt_2 ; not a valid file !
15968      <1>      ;
15969 000110C3 39C8 <1>      cmp     eax, ecx ; same first cluster ?
15970      <1>      ;je      short uflmdt_3 ; yes, it is OK !!!
15971      <1>      ; 23/07/2022
15972 000110C5 756D <1>      jne     short uflmdt_2
15973      <1>      ;
15974      <1>      ;;;
15975      <1>      ; 03/09/2024
15976 000110C7 D0E3 <1>      shl     bl, 1 ; *2
15977      <1>      ; 27/08/2024
15978 000110C9 21C0 <1>      and     eax, eax
15979 000110CB 751B <1>      jnz     short uflmdt_3 ; not empty file
15980 000110CD 8705[50830100] <1>      xchg    eax, [setfclust]
15981 000110D3 09C0 <1>      or      eax, eax
15982 000110D5 7411 <1>      jz      short uflmdt_3
15983      <1>      ; 03/09/2024
15984 000110D7 8983[64830100] <1>      mov     [ebx+OF_FCLUSTER], eax
15985      <1>      ;
15986 000110DD 6689461A <1>      mov     [esi+DirEntry_FstClusLO], ax
15987 000110E1 C1E810 <1>      shr     eax, 16
15988 000110E4 66894614 <1>      mov     [esi+DirEntry_FstClusHI], ax
15989      <1>      ;;;
15990      <1>      ;
15991      <1>      uflmdt_3:
15992      <1>      ; Update directory entry
15993      <1>      ; 03/09/2024
15994      <1>      ; 26/10/2016
15995      <1>      ;shl     bl, 1 ; *2
15996 000110E8 8883[E4840100] <1>      mov     eax, [ebx+OF_SIZE] ; file size
15997 000110EE 89461C <1>      mov     [esi+DirEntry_FileSize], eax
15998      <1>      ;
15999 000110F1 E8F69BFFFF <1>      call    convert_current_date_time
16000      <1>      ; OUTPUT -> DX = Date in dos dir entry format
16001      <1>      ;      AX = Time in dos dir entry format
16002 000110F6 66894616 <1>      mov     [esi+DirEntry_WrtTime], ax
16003 000110FA 66895618 <1>      mov     [esi+DirEntry_WrtDate], dx
16004 000110FE 66895612 <1>      mov     [esi+DirEntry_LastAccDate], dx
16005 00011102 C605[897E0100]02 <1>      mov     byte [DirBuff_ValidData], 2
16006      <1>      ;call    save_directory_buffer
16007      <1>      ;retn
16008      <1>      ; 23/07/2022
16009 00011109 E9739CFFFF <1>      jmp     save_directory_buffer
16010      <1>      ;
16011      <1>      uflmdt_4:
16012      <1>      ; Directory buffer sector read&write
16013      <1>      ; 23/10/2016
16014      <1>      ; 27/08/2024
16015      <1>      ;mov     al, [esi+LD_FATType]
16016      <1>      uflmdt_5:
16017 0001110E BB[8C940100] <1>      mov     ebx, rw_buffer ; Common r/w sector buffer addr
16018      <1>      ;
16019      <1>      ; 19/12/2025
16020      <1>      %if 0
16021      <1>      and     al, al ; 0 = Singlix FS
16022      <1>      jz      short uflmdt_11
16023      <1>      %endif
16024      <1>      ;
16025      <1>      ;uflmdt_12:
16026 00011113 21D2 <1>      and     edx, edx
16027 00011115 7527 <1>      jnz     short uflmdt_9
16028      <1>      ;
16029 00011117 3C02 <1>      cmp     al, 2 ; 3 = FAT32
16030 00011119 7720 <1>      ja      short uflmdt_8
16031      <1>      ;
16032 0001111B 89F8 <1>      mov     eax, edi ; directory entry index number
16033      <1>      ;shr     ax, 4 ; 16 entries per sector
16034      <1>      ; 23/07/2022
16035 0001111D C1E804 <1>      shr     eax, 4
16036 00011120 034664 <1>      add     eax, [esi+LD_ROOTBegin]
16037      <1>      ; eax = root directory sector
16038      <1>      uflmdt_6:
16039      <1>      push    eax ; * ; disk sector address
16040      <1>      push    ecx ; first cluster
16041 00011125 B901000000 <1>      mov     ecx, 1
16042      <1>      ; ecx = sector count
16043 0001112A E8B4080000 <1>      call    disk_read
16044 0001112F 59 <1>      pop     ecx
16045 00011130 7320 <1>      jnc     short uflmdt_10
16046 00011132 58 <1>      pop     eax ; *
16047      <1>      uflmdt_7:
16048 00011133 C3 <1>      retn
16049      <1>      ;
16050      <1>      uflmdt_2:
16051      <1>      ; save directory buffer if has modified/changed sign
16052      <1>      ; (It is good to save dir buff even if the searched
16053      <1>      ; directory entry is not found !?)
16054 00011134 E8489CFFFF <1>      call    save_directory_buffer
16055 00011139 F9 <1>      stc     ; update failed
16056 0001113A C3 <1>      retn
16057      <1>      ;
16058      <1>      ; 19/12/2025
16059      <1>      %if 0
16060      <1>      uflmdt_11:
16061      <1>      ; 24/10/2016
16062      <1>      ; Update last modification date & time of a file
16063      <1>      ; on a disk with Singlix File System.
16064      <1>      ;
16065      <1>      ; (Method: Read the FDT -File Description Table-
16066      <1>      ; sector of the file and update the lmdt data fields,
16067      <1>      ; then write FDT sector to the disk.
16068      <1>      ; /// It is easy but there is compatibility buffer
16069      <1>      ; method also for changing directory entry data and
16070      <1>      ; also there are some programming issues for Singlix
16071      <1>      ; file system (TRFS), which are not completed yet!)
16072      <1>      ;
16073      <1>      ; Not ready yet ! (24/10/2016)
16074      <1>      ; /// Temporary code for error return ! ///

```

```

16075 <1> xor eax, eax
16076 <1> stc
16077 <1> retn
16078 <1> %endif
16079 <1>
16080 <1> uflmdt_8:
16081 0001113B 8B5632 <1> mov edx, [esi+LD_BPB+FAT32_RootFClust]
16082 <1> uflmdt_9:
16083 0001113E 83FA02 <1> cmp edx, 2
16084 00011141 72F0 <1> jb short uflmdt_7 ; invalid, nothing to do
16085 <1>
16086 <1> ;sub edx, 2
16087 <1> ; 30/07/2022
16088 00011143 4A <1> dec edx
16089 00011144 4A <1> dec edx
16090 00011145 89D0 <1> mov eax, edx
16091 00011147 0FB65613 <1> movzx edx, byte [esi+LD_BPB+SecPerClust]
16092 00011148 F7E2 <1> mul edx
16093 0001114D 034668 <1> add eax, [esi+LD_DATABegin]
16094 <1> ; eax = sub directory (data) sector
16095 00011150 EBD1 <1> jmp short uflmdt_6
16096 <1>
16097 <1> uflmdt_10:
16098 <1> ; Directory sector buffer is ready here!
16099 <1> ; OF_FCLUSTER must be compared/verified
16100 <1> ; edi = dir entry index number (<= 2047)
16101 00011152 6683E70F <1> and di, 0Fh ; 16 entries per sector
16102 <1> ;shl di, 5 ; dir entry index * 32
16103 <1> ; 23/07/2022
16104 00011156 C1E705 <1> shl edi, 5
16105 00011159 81C7[8C940100] <1> add edi, rw_buffer
16106 <1> ;
16107 0001115F F6470B18 <1> test byte [edi+DirEntry_Attr], 18h ; Vol & Dir
16108 00011163 75CF <1> jnz short uflmdt_2 ; not a valid file !
16109 00011165 668B5714 <1> mov dx, [edi+DirEntry_FstClusHI]
16110 00011169 C1E210 <1> shl edx, 16
16111 0001116C 668B571A <1> mov dx, [edi+DirEntry_FstClusLO]
16112 00011170 39CA <1> cmp edx, ecx ; same first cluster ?
16113 00011172 75C0 <1> jne short uflmdt_2 ; no !?
16114 <1>
16115 <1> ;;;
16116 <1> ; 27/08/2024
16117 00011174 21D2 <1> and edx, edx
16118 00011176 7515 <1> jnz short uflmdt_12 ; not empty file
16119 00011178 8715[50830100] <1> xchg edx, [setfclust]
16120 0001117E 09D2 <1> or edx, edx
16121 00011180 740B <1> jz short uflmdt_12
16122 00011182 6689571A <1> mov [edi+DirEntry_FstClusLO], dx
16123 00011186 C1EA10 <1> shr edx, 16
16124 00011189 66895714 <1> mov [edi+DirEntry_FstClusHI], dx
16125 <1> uflmdt_12:
16126 <1> ;;;
16127 <1>
16128 <1> ; Update directory entry
16129 0001118D E85A9BFFFF <1> call convert_current_date_time
16130 <1> ; OUTPUT -> DX = Date in dos dir entry format
16131 <1> ; AX = Time in dos dir entry format
16132 00011192 66894716 <1> mov [edi+DirEntry_WrtTime], ax
16133 00011196 66895718 <1> mov [edi+DirEntry_WrtDate], dx
16134 0001119A 66895712 <1> mov [edi+DirEntry_LastAccDate], dx
16135 <1>
16136 0001119E 58 <1> pop eax ; *
16137 <1>
16138 0001119F BB[8C940100] <1> mov ebx, rw_buffer ; Common r/w sector buffer addr
16139 000111A4 B901000000 <1> mov ecx, 1
16140 <1> ; esi = logical dos description table address
16141 <1> ; eax = disk sector number/address (LBA)
16142 <1> ; ecx = sector count
16143 <1> ; ebx = buffer address
16144 000111A9 E826080000 <1> call disk_write
16145 000111AE 7284 <1> jc short uflmdt_2
16146 <1> ;uflmdt_12:
16147 <1> ; save directory buffer if has modified/changed sign
16148 <1> ;call save_directory_buffer
16149 <1> ;retn
16150 <1> ; 23/07/2022
16151 000111B0 E9CC9BFFFF <1> jmp save_directory_buffer
16152 <1>
16153 <1> sysalloc:
16154 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
16155 <1> ; 14/10/2017
16156 <1> ; 20/08/2017 - 01/09/2017
16157 <1> ; 20/02/2017 - 04/03/2017 - 15/05/2017
16158 <1> ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)
16159 <1> ; (TRDOS 386 feature only!)
16160 <1> ;
16161 <1> ; Allocate Contiguous Memory Block/Pages (for user)
16162 <1> ; (System call for DMA Buffer allocation etc.)
16163 <1> ;
16164 <1> ; INPUT ->
16165 <1> ; EBX = virtual address (for user)
16166 <1> ; (Physical memory block/aperture
16167 <1> ; will be mapped to this virtual address)
16168 <1> ; ECX = Byte Count
16169 <1> ; (will be rounded up to page border)
16170 <1> ; If ECX = 0
16171 <1> ; System call will return with an error (cf=1)
16172 <1> ; but ECX will contain maximum size of
16173 <1> ; available memory aperture and physical
16174 <1> ; (beginning) address of that aperture
16175 <1> ; (which have maximum size) will be in EAX.
16176 <1> ; EDX = Upper limit of the requested physical memory
16177 <1> ; block/pages.
16178 <1> ; (The last byte address of the memory aperture
16179 <1> ; must not be equal to or above this limit.)
16180 <1> ; If EDX = 0
16181 <1> ; there is NOLIMIT !
16182 <1> ; If EDX = 0FFFFFFFFh (-1)
16183 <1> ; ESI = Lower Limit !
16184 <1> ; (Beginning of the block must not be 'less'
16185 <1> ; than this.) (Must be equal to or above...)
16186 <1> ; EDI = Upper Limit !
16187 <1> ; (End of the block must be !less! than this)
16188 <1> ; (The last byte addr of the memory aperture
16189 <1> ; must not be equal to or above this limit.)
16190 <1> ;
16191 <1> ; OUTPUT ->
16192 <1> ; If CF = 0
16193 <1> ; EAX = Physical address of the allocated memory block
16194 <1> ; ECX = Allocated bytes (as rounded up to page borders)
16195 <1> ; EBX = Virtual address (as rounded up)
16196 <1> ; IF CF = 1
16197 <1> ; Requested (size of) Memory block could not be
16198 <1> ; allocated to the user!

```

```

16199 <1> ; IF CF = 1 & EAX = 0 (Insufficient memory error!)
16200 <1> ; ECX = Total number of free bytes
16201 <1> ; (not size of available contiguous bytes!)
16202 <1> ;
16203 <1> ; If CF = 1 & EAX > 0
16204 <1> ; there is not a memory aperture with requested size
16205 <1> ; but total free mem is not less than requested size.
16206 <1> ; EAX = Physical addr of available memory aperture
16207 <1> ; with max size
16208 <1> ; (but it doesn't fit to the conditions!)
16209 <1> ; ECX = Size of available memory aperture in bytes.
16210 <1> ; If CF = 1 -> EAX = 0FFFFFFFh
16211 <1> ; Conditions/Parameters are wrong !
16212 <1> ; ECX is same with input value.
16213 <1> ;
16214 <1> ; Note: Previously allocated pages will be deallocated if
16215 <1> ; new allocation conditions are met.
16216 <1> ;
16217 <1> ; Note: u.break control may be included in future versions
16218 <1> ;
16219 000111B5 31C0 <1> xor eax, eax ; 0
16220 <1> ; 14/10/2017
16221 000111B7 4A <1> dec edx ; is there a limit ?
16222 000111B8 7810 <1> js short sysalloc_1 ; 0 -> 0FFFFFFFh -> NO LIMIT
16223 000111BA 42 <1> inc edx ; > 0
16224 <1> ; Check upper address limit
16225 <1> ; (round up to page borders)
16226 000111BB 81C1FF0F0000 <1> add ecx, PAGE_SIZE-1 ; 4095
16227 000111C1 6681E100F0 <1> and cx, ~PAGE_OFF ; not 4095
16228 000111C6 39CA <1> cmp edx, ecx ; upper limit - block size
16229 000111C8 7224 <1> jb short sysalloc_err
16230 <1> sysalloc_1:
16231 <1> ; EAX = Beginning address (physical)
16232 <1> ; EAX = 0 -> Allocate mem block from the 1st proper aperture
16233 <1> ; ECX = Number of bytes to be allocated
16234 000111CA E86A4BFFFF <1> call allocate_memory_block
16235 000111CF 721D <1> jc short sysalloc_err
16236 <1> ; 01/09/2017
16237 000111D1 29C2 <1> sub edx, eax ; upper limit address - beginning address
16238 000111D3 760F <1> jna short sysalloc_3 ; begin addr not less than the limit
16239 000111D5 39CA <1> cmp edx, ecx
16240 000111D7 720B <1> jb short sysalloc_3 ; end address overs the limit
16241 <1> sysalloc_2:
16242 <1> ; EAX = Beginning (physical) addr of the allocated mem block
16243 <1> ; ECX = Num of allocated bytes (rounded up to page borders)
16244 000111D9 50 <1> push eax ; * ; 04/03/2017
16245 <1> ; Here, requested contiguous memory pages have been allocated
16246 <1> ; on Memory Allocation Table but user's page directory
16247 <1> ; and page tables have not been updated yet!
16248 000111DA 51 <1> push ecx ; **
16249 <1> ; ebx = virtual address (will be rounded up to page border)
16250 <1> ; ecx = number of bytes to be deallocated
16251 <1> ; will be adjusted to ebx+ecx round down - ebx round up
16252 000111DB E8B14EFFFF <1> call deallocate_user_pages
16253 000111E0 731F <1> jnc short sysalloc_4 ; EAX = Deallocated memory bytes
16254 000111E2 59 <1> pop ecx ; **
16255 000111E3 58 <1> pop eax ; *
16256 <1> sysalloc_3:
16257 <1> ; error !
16258 <1> ; restore Memory Allocation Table Content
16259 000111E4 E86A4DFFFF <1> call deallocate_memory_block
16260 000111E9 31C0 <1> xor eax, eax ; 0
16261 000111EB 48 <1> dec eax ; 0FFFFFFFh ; 15/05/2017
16262 000111EC EB09 <1> jmp short sysalloc_wrong
16263 <1> sysalloc_err:
16264 000111EE 8B2D[508E0100] <1> mov ebp, [u.usp] ; ebp points to user's registers
16265 000111F4 894D18 <1> mov [ebp+24], ecx ; return to user with ecx value
16266 <1> sysalloc_wrong:
16267 <1> ; eax = 0FFFFFFFh
16268 000111F7 A3[548E0100] <1> mov [u.r0], eax
16269 000111FC E95DB8FFFF <1> jmp error
16270 <1> sysalloc_4:
16271 00011201 8B2D[508E0100] <1> mov ebp, [u.usp] ; ebp points to user's registers
16272 00011207 894518 <1> mov [ebp+24], eax ; return to user with ecx value
16273 0001120A 895D10 <1> mov [ebp+16], ebx ; new value of ebx (rounded up)
16274 0001120D 89C1 <1> mov ecx, eax ; byte count (from 'deallocate_user_pages')
16275 0001120F 5A <1> pop edx ; ** ; discard (another) byte count
16276 00011210 58 <1> pop eax ; *
16277 00011211 A3[548E0100] <1> mov [u.r0], eax ; physical address
16278 <1>
16279 00011216 51 <1> push ecx ; 20/08/2017
16280 <1>
16281 <1> ; write newly allocated contiguous (physical) pages
16282 <1> ; on page dir and page tables of current user/process
16283 <1> ; as PRESENT, USER, WRITABLE
16284 <1> ; (then clear allocated pages)
16285 00011217 E85D4FFFFF <1> call allocate_user_pages
16286 <1> ; jnc sysret ; OK! return to process with success...
16287 <1>
16288 <1> ; 20/08/2017 ('sysdma' modification)
16289 0001121C 59 <1> pop ecx
16290 0001121D A1[548E0100] <1> mov eax, [u.r0] ; physical address (of the block)
16291 <1>
16292 00011222 7219 <1> jc short sysalloc_6
16293 <1>
16294 00011224 833D[188D0100]FF <1> cmp dword [dma_addr], 0FFFFFFFh ; -1
16295 <1> ; jb sysret
16296 <1> ; 23/07/2022
16297 0001122B 720B <1> jb short sysalloc_5 ; jmp sysret
16298 <1>
16299 0001122D A3[188D0100] <1> mov [dma_addr], eax ; save dma address for sysdma
16300 00011232 890D[1C8D0100] <1> mov [dma_size], ecx ; save dma buff size for sysdma
16301 <1> sysalloc_5:
16302 00011238 E941B8FFFF <1> jmp sysret
16303 <1>
16304 <1> sysalloc_6:
16305 <1>
16306 <1> ; unexpected error ! insufficient memory !? conflict !?
16307 <1> ; (!?there is not a free page for a new page table?!!)
16308 <1> ; we need to terminate process with error message !!!
16309 <1>
16310 0001123D 8B2D[508E0100] <1> mov ebp, [u.usp] ; ebp points to user's registers
16311 00011243 8B4D18 <1> mov ecx, [ebp+24] ; byte count
16312 <1>
16313 <1> ; 20/08/2017
16314 <1> ; mov eax, [u.r0] ; physical address (of the block)
16315 <1>
16316 <1>
16317 <1> ; restore Memory Allocation Table Content
16318 00011246 E8084DFFFF <1> call deallocate_memory_block
16319 <1>
16320 0001124B 803D[1E680000]03 <1> cmp byte [CRT_MODE], 3 ; 80x25 text mode?
16321 00011252 7407 <1> je short sysalloc_7 ; yes
16322 <1> ; Current mode is VGA (or CGA graphics) mode,

```

```

16323      <1>      ; We need to return to text mode for displaying
16324      <1>      ; error message just before 'sysexit'.
16325      00011254 B003      <1>      mov     al, 3
16326      00011256 E82E09FFFF <1>      call    _set_mode
16327      <1>      sysalloc_7:
16328      0001125B BE[E8310100] <1>      mov     esi, beep_Insufficient_Memory ; error message
16329      00011260 E8455BFFFF <1>      call    print_msg ; print/display the message
16330      00011265 B801000000 <1>      mov     eax, 1 ; ax=1 is needed for 'sysexit' procedure
16331      0001126A E94DB9FFFF <1>      jmp     sysexit ; and terminate the process !
16332      <1>
16333      <1>      sysdalloc:
16334      <1>      ; 19/02/2017 - TRDOS 386 (TRDOS v2.0)
16335      <1>      ; (TRDOS 386 feature only!)
16336      <1>
16337      <1>      ; Deallocate Memory Block/Pages (for user)
16338      <1>      ; (Complementary call for sysalloc.)
16339      <1>
16340      <1>      ; INPUT ->
16341      <1>      ;     EBX = Virtual address (for user)
16342      <1>      ;     (will be rounded up to page border)
16343      <1>      ;     ECX = Byte Count
16344      <1>      ;     (will be adjusted to page borders)
16345      <1>      ;     If ICX = 0
16346      <1>      ;     nothing to do
16347      <1>      ;     If EBX + ECX > User's ESP
16348      <1>      ;     nothing to do
16349      <1>
16350      <1>      ; Note: u.break control may be included in future versions
16351      <1>
16352      <1>      ; OUTPUT ->
16353      <1>      ;     If CF = 0
16354      <1>      ;     EAX = Deallocated memory bytes
16355      <1>      ;     EBX = Virtual address (as rounded up)
16356      <1>      ;     IF CF = 1
16357      <1>      ;     EAX = 0
16358      <1>
16359      <1>      ; Note: Main purpose of this call is to deallocate/release
16360      <1>      ;     previously allocated (physically) contiguous memory
16361      <1>      ;     pages but beginning (virtual) address may not be
16362      <1>      ;     followed by physically contiguous pages. So, this
16363      <1>      ;     system call will deallocate user's virtually
16364      <1>      ;     contiguous memory pages. Also, there is not any
16365      <1>      ;     objections to use this system call without sysalloc
16366      <1>      ;     system call; only possible objection is to lost data
16367      <1>      ;     within user's memory space, if the beginning address
16368      <1>      ;     and size is not proper.
16369      <1>
16370      <1>      ; Note: Empty page tables will not be deallocated!!!
16371      <1>      ;     (they will be deallocated at process termination)
16372      <1>
16373      <1>      ; Note: When the program terminates itself or when it is
16374      <1>      ;     terminated by operating system kernel, all allocated
16375      <1>      ;     memory pages will be deallocated during termination
16376      <1>      ;     stage. So, 'sysdalloc' is not necessary except
16377      <1>      ;     forgiving memory block to other programs/processes.
16378      <1>
16379      0001126F 8B15[4C8E0100] <1>      mov     edx, [u.sp]
16380      00011275 8B420C      <1>      mov     eax, [edx+12] ; user's stack pointer
16381      00011278 29C8      <1>      sub     eax, ecx ; esp - byte count
16382      0001127A 24FC      <1>      and     al, 0FCh ; dword alignment
16383      0001127C 39D8      <1>      cmp     eax, ebx
16384      0001127E 7220      <1>      jb     short sysdalloc_err ; deallocation overlaps with stack
16385      <1>
16386      00011280 31C0      <1>      xor     eax, eax
16387      00011282 21C9      <1>      and     ecx, ecx
16388      00011284 7407      <1>      jz     short sysdalloc_2
16389      <1>
16390      00011286 E8064EFFFF <1>      call    deallocate_user_pages
16391      0001128B 7213      <1>      jc     short sysdalloc_err
16392      <1>
16393      <1>      sysdalloc_2:
16394      0001128D A3[548E0100] <1>      mov     [u.r0], eax
16395      00011292 8B2D[508E0100] <1>      mov     ebp, [u.usp]
16396      00011298 895D10      <1>      mov     [ebp+16], ebx ; new value of ebx
16397      0001129B E9DEB7FFFF <1>      jmp     sysret
16398      <1>
16399      <1>      sysdalloc_err:
16400      000112A0 A3[548E0100] <1>      mov     [u.r0], eax ; 0
16401      000112A5 E9B4B7FFFF <1>      jmp     error
16402      <1>
16403      <1>      syscallback:
16404      <1>      ; SYS CALLBACK
16405      <1>      ; 23/07/2022 - TRDOS 386 v2.0.5
16406      <1>      ; 03/08/2020
16407      <1>      ; 16/04/2017
16408      <1>      ; 14/04/2017
16409      <1>      ; 13/04/2017
16410      <1>      ; 28/02/2017
16411      <1>      ; 26/02/2017
16412      <1>      ; 24/02/2017
16413      <1>      ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)
16414      <1>      ; (TRDOS 386 feature only!)
16415      <1>
16416      <1>      ; Link or unlink IRQ callback service to/from user (ring 3)
16417      <1>
16418      <1>      ; INPUT ->
16419      <1>      ;     BL = IRQ number (Hardware interrupt request number)
16420      <1>      ;     (0 to 15 but IRQ 0,1,2,6,8,14,15 are prohibited)
16421      <1>      ;     IRQ numbers 3,4,5,7,9,10,11,12,13 are valid
16422      <1>      ;     (numbers >15 are invalid)
16423      <1>
16424      <1>      ;     BH = 0 = Unlink IRQ (in BL) from user (ring 3) service
16425      <1>      ;     1 = Link IRQ by using Signal Response Byte method
16426      <1>      ;     2 = Link IRQ by using Callback service method
16427      <1>      ;     3 = Link IRQ by using Auto Increment S.R.B. method
16428      <1>      ;     >3 = invalid
16429      <1>
16430      <1>      ;     CL = Signal Return/Response Byte value
16431      <1>
16432      <1>      ;     If BH = 3, kernel will put a counter value ; 03/08/2020
16433      <1>      ;     (into the S.R.B. addr)
16434      <1>      ;     between 0 to 255. (start value = CL+1)
16435      <1>
16436      <1>      ; NOTE: counter value, for example: even and odd numbers
16437      <1>      ;     may be used for -audio- DMA buffer switch
16438      <1>      ;     within double buffer method, etc.
16439      <1>
16440      <1>      ;     EDX = Signal return (Response) byte address
16441      <1>      ;     - or -
16442      <1>      ;     Interrupt/Callback service/routine address
16443      <1>      ;
16444      <1>      ;     (virtual address in user's memory space)
16445      <1>
16446      <1>      ; OUTPUT ->

```

```

16447 <1> ; CF = 0 & EAX = 0 -> Successful setting
16448 <1> ; CF = 1 & EAX > 0 -> IRQ is prohibited or locked
16449 <1> ; by another process
16450 <1> ; eax = ERR_PERM_DENIED -> prohibited or locked
16451 <1> ; eax = ERR_INV_PARAMETER ->
16452 <1> ; invalid parameter/option or bad address
16453 <1> ;
16454 <1> ; NOTE: Timer callbacks are set by using 'systimer'
16455 <1> ; system call (IRQ 0, PIT and IRQ 8, RTC)
16456 <1> ;
16457 <1> ; Direct keyboard access is performed by using
16458 <1> ; Keyboard Interrupt (INT 32h)
16459 <1> ;
16460 <1> ; It is prohibited here because:
16461 <1> ; 1) Signal Response Byte method has not advantage
16462 <1> ; against INT 32h, function AH = 1. Also,
16463 <1> ; keyboard service interrupt will return with
16464 <1> ; ascii and scan codes (AL, AH) while
16465 <1> ; SRB method has only 1 byte space for ascii code
16466 <1> ; or scan code. One byte signal response is used
16467 <1> ; for ensuring very simple and very fast
16468 <1> ; virtual to physical memory address conversion
16469 <1> ; without any memory page crossover risk.
16470 <1> ; (Otherwise double page conversion or word
16471 <1> ; alignment would be needed.)
16472 <1> ; 2) Badly written user code (callback code)
16473 <1> ; can prevent keyboard and timesharing functions
16474 <1> ; of the operating system via continuous and long
16475 <1> ; keyboard event handling by callback service.
16476 <1> ; (It can cause to lose immediate keystroke
16477 <1> ; response from hardware to user.)
16478 <1> ; 3) If user will check any keyboard events, 'getkey'
16479 <1> ; (or 'getchar') must have more priority than other
16480 <1> ; (video etc.) events because only control ability
16481 <1> ; on a procedural infinite loop is a keyboard or
16482 <1> ; mouse event. So user can use keyboard function
16483 <1> ; at the end or at the beginning of a loop.
16484 <1> ; In this case, INT 32h is used for that purpose
16485 <1> ; and timer interrupt etc. callbacks can be used
16486 <1> ; for dynamic and synchronized data refresh/transfer
16487 <1> ; while cpu is in a static loop (without polling).
16488 <1> ; Keyboard Int callback is not more useful because
16489 <1> ; already a manual check (a key is pressed or not)
16490 <1> ; can be performed (via INT 32h, AH = 1) efficiently
16491 <1> ; in a loop to prevent a locked infinitive loop.
16492 <1> ;
16493 <1> ; Disk IRQs (6,14,15) have been phohibited from ring 3
16494 <1> ; callback because, disk operations (file system services
16495 <1> ; etc.) are independent from user program, for fast disk r/w.
16496 <1> ; They are not more useful at ring 3 while they are in use
16497 <1> ; by standard diskio functions which are mandatory part of
16498 <1> ; (monolithic) OS kernel and mainprog command interpreter.
16499 <1> ; INT 33h diskio functions are enough for user level disk
16500 <1> ; r/w.
16501 <1> ;
16502 <1> ; TRDOS 386 - IRQ CALLBACK structures (parameters):
16503 <1> ;
16504 <1> ; [u.irqlock] = 1 word, IRQ flags (0-15) that indicates
16505 <1> ; which IRQs are locked by (that) user.
16506 <1> ; Lock and unlock (by user) will change
16507 <1> ; these flags or 'terminate process' (sysexit)
16508 <1> ; will clear these flags and unlock those IRQs.
16509 <1> ;
16510 <1> ; Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15
16511 <1> ;
16512 <1> ; IRQ(x).owner : 1 byte, user, [u.uno], 0 = free (unlocked)
16513 <1> ;
16514 <1> ; IRQ(x).method : 1 byte for callback method & status
16515 <1> ; 0 = Signal Response Byte method
16516 <1> ; 1 = Callback service method
16517 <1> ; >1 = invalid for current 'syscallback'.
16518 <1> ; or(+) 80h = IRQ is in use by system (ring 0)
16519 <1> ; function (audio etc.) or
16520 <1> ; a device driver.
16521 <1> ; (system function will ignore the lock/owner)
16522 <1> ;
16523 <1> ; IRQ(x).srb : 1 byte, Signal Return/Response byte value
16524 <1> ; (a fixed value by user or a counter value
16525 <1> ; from 0 to 255, which is increased by every
16526 <1> ; interrupt just before putting it into
16527 <1> ; the Signal Response byte address
16528 <1> ; (This is not used in callback serv method)
16529 <1> ;
16530 <1> ; IRQ(x).addr : 1 dword
16531 <1> ; Signal Response Byte address (physical)
16532 <1> ; -or-
16533 <1> ; callback service address (virtual)
16534 <1> ;
16535 <1> ; IRQ(x).dev : 1 byte
16536 <1> ; 0 = Default device or kernel function
16537 <1> ; -or-
16538 <1> ; 1-255 = Assigned device driver number
16539 <1> ;
16540 <1> ; (x) = 3,4,5,7,9,10,11,12,13
16541 <1> ;
16542 <1> ; NOTE: If user's process/program calls the kernel (INT 40h)
16543 <1> ; while it is already running in a (ring 3) callback
16544 <1> ; service, kernel will force (convert) system call to
16545 <1> ; 'sysrele' (sys release). So, this feature provides
16546 <1> ; easy and simple usage of callback services without
16547 <1> ; falling into deepless <please 'callback me' then
16548 <1> ; let me 'callback you'> cycles! (User must return
16549 <1> ; from callback service by using 'sysrele' system
16550 <1> ; call, without a significant delay. Otherwise user
16551 <1> ; process/program may be late to catch the next event
16552 <1> ; within same callback purpose.
16553 <1> ;
16554 <1> ;
16555 <1> ;
16556 <1> ; xor al, al ; the caller is 'syscallback' sign/flag
16557 <1> ; call set_irq_callback_service
16558 <1> ; ; 16/04/2017
16559 <1> ; mov [u.r0], eax
16560 <1> ; jnc sysret
16561 <1> ; ; 23/07/2022
16562 <1> ; jc short syscallback_err
16563 <1> ; jmp sysret
16564 <1> ; syscallback_err:
16565 <1> ; mov dword [u.error], eax
16566 <1> ; jmp error
16567 <1> ;
16568 <1> ; sysfpstat:
16569 <1> ; ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)
16570 <1> ; ; (TRDOS 386 feature only!)

```

```

16571 <1> ;
16572 <1> ; Set or reset FPU registers save/restore option (for user)
16573 <1> ; (during software task switching, wswap-rswap)
16574 <1> ;
16575 <1> ; INPUT ->
16576 <1> ; BL = 0 -> reset
16577 <1> ; BL = 1 -> set (FPU register will be saved and restored)
16578 <1> ;
16579 <1> ; OUTPUT ->
16580 <1> ; cf = 0 -> no error, FPU is ready...
16581 <1> ; (EAX = 0)
16582 <1> ; Cf = 1 -> error, 80387 FPU is not ready !
16583 <1> ; (EAX = 0FFFFFFFh)
16584 <1> ;
16585 000112C7 31C0 <1> xor     eax, eax
16586 000112C9 803D[60830100]00 <1> cmp     byte [fpready], 0
16587 000112D0 7613 <1> jna     short sysfpstat_err
16588 <1> ;
16589 000112D2 80E301 <1> and     bl, 1 ; use BIT 0 only !
16590 000112D5 881D[D38E0100] <1> mov     [u.fpsave], bl
16591 000112DB A3[548E0100] <1> mov     [u.r0], eax ; 0
16592 000112E0 E999B7FFFF <1> jmp     sysret
16593 <1> ;
16594 <1> sysfpstat_err:
16595 000112E5 48 <1> dec     eax ; 0FFFFFFFh
16596 000112E6 A3[548E0100] <1> mov     [u.r0], eax ; -1
16597 000112EB E96EB7FFFF <1> jmp     error
16598 <1> ;
16599 <1> sysdelete: ; Delete (Remove, Unlink) File
16600 <1> ; 29/12/2017 (TRDOS 386 = TRDOS v2.0)
16601 <1> ;
16602 <1> ; INPUT ->
16603 <1> ; EBX = File name (ASCII string) address
16604 <1> ; OUTPUT ->
16605 <1> ; cf = 0 -> eax = 0
16606 <1> ; cf = 1 -> Error code in AL
16607 <1> ;
16608 <1> ; Modified Registers: EAX (at the return of system call)
16609 <1> ;
16610 <1> ;
16611 000112F0 89DE <1> mov     esi, ebx
16612 <1> ; file name is forced, change directory as temporary
16613 <1> ; mov     ax, 1
16614 <1> ; mov     [FFF_Valid], ah ; 0 ; reset
16615 <1> ; call    set_working_path
16616 000112F2 E8CE090000 <1> call    set_working_path_x
16617 000112F7 731D <1> jnc     short sysdelete_1
16618 <1> ;
16619 000112F9 21C0 <1> and     eax, eax ; 0 -> Bad Path!
16620 000112FB 7505 <1> jnz     short sysdelete_err
16621 <1> ; eax = 0
16622 <1> sysdelete_path_err:
16623 000112FD B813000000 <1> mov     eax, ERR_INV_PATH_NAME ; 'bad path name !'
16624 <1> sysdelete_err:
16625 00011302 A3[548E0100] <1> mov     [u.r0], eax
16626 00011307 A3[C08E0100] <1> mov     [u.error], eax
16627 0001130C E8890A0000 <1> call    reset_working_path
16628 00011311 E948B7FFFF <1> jmp     error
16629 <1> sysdelete_1:
16630 <1> ; mov     esi, FindFile_Name
16631 00011316 66B80018 <1> mov     ax, 1800h ; Only files
16632 0001131A E84B76FFFF <1> call    find_first_file
16633 0001131F 72E1 <1> jc      short sysdelete_err
16634 <1> sysdelete_2:
16635 <1> ; check file attributes
16636 <1> ;
16637 <1> ; test    bl, 17 ; system, hidden, readonly, directory
16638 00011321 F6C307 <1> test     bl, 7 ; system, hidden, readonly
16639 00011324 7407 <1> jz      short sysdelete_3
16640 <1> ;
16641 00011326 B80B000000 <1> mov     eax, ERR_FILE_ACCESS ; 11 = 'permission denied !'
16642 0001132B EBD5 <1> jmp     short sysdelete_err
16643 <1> sysdelete_3:
16644 0001132D 6621D2 <1> and     dx, dx ; Ambiguous filename chars used sign (DX>0)
16645 00011330 7407 <1> jz      short sysdelete_4
16646 00011332 B81A000000 <1> mov     eax, ERR_INV_FILE_NAME ; 26 = 'invalid file name !'
16647 00011337 EBC9 <1> jmp     short sysdelete_err
16648 <1> sysdelete_4:
16649 <1> ; mov     bh, [LongName_EntryLength]
16650 00011339 883D[CA800100] <1> mov     [DelFile_LNELL], bh ; Long name entry length (if > 0)
16651 <1> ; edi = Directory Entry Offset (DirBuff)
16652 <1> ; esi = Directory Entry (FFF Structure)
16653 0001133F E8559DFFFF <1> call    remove_file
16654 00011344 72BC <1> jc      short sysdelete_err
16655 <1> sysrmdir_5:
16656 00011346 31C0 <1> xor     eax, eax ; 0
16657 00011348 A3[548E0100] <1> mov     [u.r0], eax
16658 <1> ; mov     [u.error], eax
16659 0001134D E8480A0000 <1> call    reset_working_path
16660 00011352 E927B7FFFF <1> jmp     sysret
16661 <1> ;
16662 <1> sysrmdir: ; Remove (Unlink) Directory
16663 <1> ; 17/07/2025 - TRDOS 386 v2.0.10
16664 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
16665 <1> ; 19/01/2021
16666 <1> ; 29/12/2017 (TRDOS 386 = TRDOS v2.0)
16667 <1> ;
16668 <1> ; INPUT ->
16669 <1> ; EBX = Pointer to directory name
16670 <1> ; OUTPUT ->
16671 <1> ; cf = 0 -> eax = 0
16672 <1> ; cf = 1 -> Error code in AL
16673 <1> ;
16674 <1> ; Modified Registers: EAX (at the return of system call)
16675 <1> ;
16676 <1> ;
16677 <1> ; 19/01/2021
16678 00011357 803D[A68E0100]00 <1> cmp     byte [u.uid], 0 ; root (super user) ?
16679 0001135E 7614 <1> jna     short sysrmdir_0
16680 <1> ;
16681 <1> ; mov     dword [u.r0], ERR_PERM_DENIED
16682 00011360 B80B000000 <1> mov     eax, ERR_PERM_DENIED ; ERR_NOT_SUPERUSER
16683 00011365 A3[548E0100] <1> mov     [u.r0], eax
16684 0001136A A3[C08E0100] <1> mov     [u.error], eax
16685 0001136F E9EAB6FFFF <1> jmp     error
16686 <1> ;
16687 <1> sysrmdir_0:
16688 00011374 89DE <1> mov     esi, ebx
16689 <1> ;
16690 <1> ;
16691 <1> ; 17/07/2025 - TRDOS 386 v2.0.10
16692 00011376 A0[62770100] <1> mov     al, [Current_Drv]
16693 0001137B A2[08810100] <1> mov     [rmdir_drv], al
16694 00011380 A1[5C770100] <1> mov     eax, [Current_Dir_FCluster]

```

```

16695 00011385 A3[04810100] <1> mov [rmdir_dir_fcluster], eax
16696 <1> ;;;;
16697 <1>
16698 <1> ; file name is forced, change directory as temporary
16699 <1> ;mov ax, 1
16700 <1> ;mov [FFF_Valid], ah ; 0 ; reset
16701 <1> ;call set_working_path
16702 0001138A E836090000 <1> call set_working_path_x
16703 0001138F 7308 <1> jnc short sysrmdir_1
16704 <1>
16705 00011391 21C0 <1> and eax, eax ; 0 -> Bad Path!
16706 00011393 7513 <1> jnz short sysrmdir_err
16707 <1> ; eax = 0
16708 <1> sysrmdir_not_found:
16709 <1> ;mov eax, ERR_DIR_NOT_FOUND ; Directory not found !
16710 <1> ; 23/07/2022
16711 00011395 B00C <1> mov al, ERR_DIR_NOT_FOUND
16712 00011397 EB0F <1> jmp short sysrmdir_err
16713 <1> ;sysrmdir_err:
16714 <1> ; mov [u.r0], eax
16715 <1> ; mov [u.error], eax
16716 <1> ; call reset_working_path
16717 <1> ; jmp error
16718 <1> sysrmdir_1:
16719 <1> ;mov esi, FindFile_Name
16720 00011399 66B81008 <1> mov ax, 0810h ; Only directories
16721 0001139D E8C875FFFF <1> call find_first_file
16722 000113A2 7318 <1> jnc short sysrmdir_2
16723 <1>
16724 <1> ; eax = 2 (File not found !)
16725 000113A4 3C02 <1> cmp al, 2 ; ERR_NOT_FOUND
16726 000113A6 74ED <1> je short sysrmdir_not_found
16727 <1> ;jmp short sysrmdir_err
16728 <1> ; 23/07/2022
16729 <1> sysrmdir_err:
16730 000113A8 A3[548E0100] <1> mov [u.r0], eax
16731 000113AD A3[C08E0100] <1> mov [u.error], eax
16732 <1> sysrmdir_8: ; 23/07/2022
16733 000113B2 E8E3090000 <1> call reset_working_path
16734 000113B7 E9A2B6FFFF <1> jmp error
16735 <1> sysrmdir_2:
16736 <1> ; check directory attributes
16737 <1>
16738 000113BC F6C307 <1> test bl, 7 ; system, hidden, readonly
16739 000113BF 7407 <1> jz short sysrmdir_3
16740 <1>
16741 000113C1 B80B000000 <1> mov eax, ERR_DIR_ACCESS ; 11 = 'permission denied !'
16742 000113C6 EBEO <1> jmp short sysrmdir_err
16743 <1> sysrmdir_3:
16744 000113C8 6621D2 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)
16745 000113CB 7407 <1> jz short sysrmdir_4
16746 <1> ;mov eax, ERR_NOT_DIR ; 'not a valid directory !'
16747 000113CD B813000000 <1> mov eax, ERR_INV_PATH_NAME ; 'bad path name !'
16748 000113D2 EBD4 <1> jmp short sysrmdir_err
16749 <1> sysrmdir_4:
16750 <1> ;mov bh, [LongName_EntryLength]
16751 000113D4 883D[CA800100] <1> mov [DelFile_LNEL], bh ; Long name entry length (if > 0)
16752 <1> ; edi = Directory Entry Offset (DirBuff)
16753 <1> ; esi = Directory Entry (FFF Structure)
16754 000113DA E8F77BFFFF <1> call delete_sub_directory
16755 <1> ;jnc sysrmdir_5
16756 <1> ; 23/07/2022
16757 000113DF 7205 <1> jc short sysrmdir_6
16758 000113E1 E960FFFFFF <1> jmp sysrmdir_5
16759 <1>
16760 <1> ; jc short sysrmdir_6
16761 <1> ;
16762 <1> ; xor eax, eax ; 0
16763 <1> ;sysrmdir_5:
16764 <1> ; mov [u.r0], eax
16765 <1> ; mov [u.error], eax
16766 <1> ; call reset_working_path
16767 <1> ; jmp sysret
16768 <1>
16769 <1> sysrmdir_6:
16770 000113E6 A3[548E0100] <1> mov [u.r0], eax
16771 000113EB A3[C08E0100] <1> mov [u.error], eax
16772 <1>
16773 000113F0 09C0 <1> or eax, eax ; EAX = 0 -> Directory not empty!
16774 000113F2 7414 <1> jz short sysrmdir_9
16775 <1>
16776 <1> ; EAX > 0 -> Error code in AL (or AX or EAX)
16777 <1>
16778 000113F4 833D[7E7E0100]01 <1> cmp dword [FAT_C]usterCounter], 1
16779 000113FB 72B5 <1> jb short sysrmdir_8
16780 <1> sysrmdir_7:
16781 <1> ; ESI = Logical DOS Drive Description Table address
16782 000113FD 66BB00FF <1> mov bx, 0FF00h ; BH = FFh -> use ESI for Drive parameters
16783 <1> ; BL = 0 -> Recalculate free cluster count
16784 00011401 E85BB1FFFF <1> call calculate_fat_freespace
16785 <1> ; 23/07/2022
16786 00011406 EBAA <1> jmp short sysrmdir_8
16787 <1> ;sysrmdir_8:
16788 <1> ; call reset_working_path
16789 <1> ; jmp error
16790 <1>
16791 <1> sysrmdir_9:
16792 00011408 A1[7E7E0100] <1> mov eax, [FAT_C]usterCounter]
16793 0001140D 09C0 <1> or eax, eax ; 0 ?
16794 <1> ;jz short sysrmdir_err
16795 <1> ; 23/07/2022
16796 0001140F 74A1 <1> jz short sysrmdir_8
16797 <1>
16798 <1> ; ESI = Logical DOS Drive Description Table address
16799 00011411 66BB01FF <1> mov bx, 0FF01h ; BH = FFh -> use ESI for Drive parameters
16800 <1> ; BL = 1 -> add free clusters
16801 00011415 E847B1FFFF <1> call calculate_fat_freespace
16802 0001141A 09C9 <1> or ecx, ecx
16803 0001141C 7494 <1> jz short sysrmdir_8 ; ecx = 0 -> OK
16804 <1> ; ecx > 0 -> Error (Recalculation is needed)
16805 0001141E EBDD <1> jmp short sysrmdir_7
16806 <1>
16807 <1> syschdir: ; Change Current (Working) Drive & Directory (for user)
16808 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
16809 <1> ;
16810 <1> ; INPUT ->
16811 <1> ; EBX = Directory name (ASCII string) address
16812 <1> ; OUTPUT ->
16813 <1> ;
16814 <1> ; cf = 0 -> eax = 0
16815 <1> ; cf = 1 -> Error code in AL
16816 <1> ;
16817 <1> ; Modified Registers: EAX (at the return of system call)
16818 <1> ; NOTE: If drive name is not included, only the working

```

```

16819      <1>      ; directory (for user, not for drive/OS) will be chanded.
16820      <1>      ; If there is a drive name (as A:, B:, C:, D: etc.)
16821      <1>      ; at the beginning of the ASCIIZ (directory) string,
16822      <1>      ; working drive and working directory (for user)
16823      <1>      ; will be changed together.
16824      <1>      ; (When the program is terminated, MainProg -internal
16825      <1>      ; shell- will reset working directory to the previous
16826      <1>      ; -current- logical drive's current directory again.)
16827      <1>
16828 00011420 89DE      <1>      mov     esi, ebx
16829      <1>      ; file name is not forced, change directory as temporary
16830 00011422 31C0      <1>      xor     eax, eax
16831      <1>      ;mov     [FFF_Valid], ah ; 0 ; reset
16832      <1>      ;call    set_working_path
16833 00011424 E8A0080000 <1>      call   set_working_path_xx
16834 00011429 731D      <1>      jnc     short syschdir_ok
16835 0001142B 21C0      <1>      and     eax, eax ; 0 -> Bad Path!
16836 0001142D 7505      <1>      jnz     short syschdir_err
16837      <1>      ; eax = 0
16838      <1>      syschdir_not_found:
16839 0001142F B80C000000 <1>      mov     eax, ERR_DIR_NOT_FOUND ; Directory not found !
16840      <1>      syschdir_err:
16841 00011434 A3[548E0100] <1>      mov     [u.r0], eax
16842 00011439 A3[C08E0100] <1>      mov     [u.error], eax
16843 0001143E E857090000 <1>      call   reset_working_path
16844 00011443 E916B6FFFF <1>      jmp     error
16845      <1>      syschdir_ok:
16846 00011448 31C0      <1>      xor     eax, eax ; 0
16847 0001144A A3[548E0100] <1>      mov     [u.r0], eax
16848      <1>      ;mov     [u.error], eax
16849 0001144F E92AB6FFFF <1>      jmp     sysret
16850      <1>
16851      <1>
16852      <1>      syschmod: ; Get & Change File (or Directory) Attributes
16853      <1>      ; 26/09/2024 - TRDOS 386 v2.0.9
16854      <1>      ; 23/07/2022 - TRDOS 386 v2.0.5
16855      <1>      ; 19/01/2021
16856      <1>      ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
16857      <1>      ;
16858      <1>      ; INPUT ->
16859      <1>      ; EBX = File/Directory (ASCIIZ) name address
16860      <1>      ; CL = New attributes (if CL < 40h)
16861      <1>      ; CL >= 40h -> Get File Attributes
16862      <1>      ; OUTPUT ->
16863      <1>      ; cf = 0 -> EAX = File attributes (in AL)
16864      <1>      ; cf = 1 -> Error code in AL
16865      <1>      ;
16866      <1>      ; Modified Registers: EAX (at the return of system call)
16867      <1>      ;
16868      <1>      ; MSDOS File Attributes: (bit value of attrib byte)
16869      <1>      ; ATTR_READ_ONLY = 01h (bit 0, 'R')
16870      <1>      ; ATTR_HIDDEN = 02h (bit 1, 'H')
16871      <1>      ; ATTR_SYSTEM = 04h (bit 2, 'S')
16872      <1>      ; ATTR_VOLUME_ID = 08h (bit 3)
16873      <1>      ; ATTR_DIRECTORY = 10h (bit 4)
16874      <1>      ; ATTR_ARCHIVE = 20h (bit 5, 'A')
16875      <1>      ; ATTR_LONG_NAME = ATTR_READONLY |
16876      <1>      ; ATTR_HIDDEN |
16877      <1>      ; ATTR_SYSTEM |
16878      <1>      ; ATTR_VOLUME_ID
16879      <1>      ; The upper two bits of attributes must be 0.
16880      <1>      ;
16881      <1>      ; Note: * If ATTR_DIRECTORY is set, only directory names
16882      <1>      ; will be searched (and S,H,R,A attributeds of
16883      <1>      ; the directory will be changed.)
16884      <1>      ; * If ATTR_VOLUME_ID is set, 'syschmod' system call
16885      <1>      ; will return with 'permission denied' error.
16886      <1>      ; * If ATTR_DIRECTORY is not set, only file names
16887      <1>      ; will be searched (and S,H,R,A attributes of the
16888      <1>      ; file will be changed.)
16889      <1>      ;
16890      <1>      ; (Only Super User can change S,H,R attributes.)
16891      <1>
16892 00011454 80F940      <1>      cmp     cl, 40h
16893 00011457 7327      <1>      jnb     short syschmod_0
16894      <1>
16895 00011459 F6C108      <1>      test    cl, 08h ; ATTR_VOLUME_ID
16896 0001145C 750E      <1>      jnz     short syschmod_perm_err
16897      <1>
16898      <1>      ; 19/01/2021
16899 0001145E 803D[A68E0100]00 <1>      cmp     byte [u.uid], 0 ; root (super user) ?
16900 00011465 7619      <1>      jna     short syschmod_0
16901      <1>
16902      <1>      ; Not super user..
16903 00011467 F6C107      <1>      test    cl, 07h ; S,H,R attributes
16904 0001146A 7414      <1>      jz      short syschmod_0
16905      <1>
16906      <1>      syschmod_perm_err:
16907      <1>      ;mov     dword [u.r0], ERR_PERM_DENIED
16908 0001146C B80B000000 <1>      mov     eax, ERR_PERM_DENIED ; 'permission denied !'
16909 00011471 A3[548E0100] <1>      mov     [u.r0], eax
16910 00011476 A3[C08E0100] <1>      mov     [u.error], eax
16911 0001147B E9DEB5FFFF <1>      jmp     error
16912      <1>
16913      <1>      syschmod_0:
16914 00011480 880D[1C810100] <1>      mov     [Attributes], cl
16915 00011486 89DE      <1>      mov     esi, ebx
16916      <1>      ; file name is forced, change directory as temporary
16917      <1>      ;mov     ax, 1
16918      <1>      ;mov     [FFF_Valid], ah ; 0 ; reset
16919      <1>      ;call    set_working_path
16920 00011488 E838080000 <1>      call   set_working_path_x
16921 0001148D 7308      <1>      jnc     short syschmod_1
16922 0001148F 21C0      <1>      and     eax, eax ; 0 -> Bad Path!
16923 00011491 751E      <1>      jnz     short syschmod_err
16924      <1>      ; eax = 0
16925      <1>      syschmod_path_not_found:
16926      <1>      ;mov     eax, ERR_INV_PATH_NAME ; 'Bad path name !'
16927      <1>      ; 23/07/2022
16928 00011493 B013      <1>      mov     al, ERR_INV_PATH_NAME
16929      <1>      ;syschmod_err:
16930      <1>      ;mov     [u.r0], eax
16931      <1>      ;mov     [u.error], eax
16932      <1>      ;call    reset_working_path
16933      <1>      ;jmp     error
16934      <1>      ; 23/07/2022
16935 00011495 EB1A      <1>      jmp     short syschmod_err
16936      <1>      syschmod_1:
16937 00011497 B008      <1>      mov     al, 08h ; Except volume labels (& long names)
16938 00011499 A0[1C810100] <1>      mov     al, [Attributes]
16939 0001149E 2410      <1>      and     al, 10h ;
16940      <1>      ;mov     esi, FindFile_Name
16941      <1>      ;mov     ax, 1800h ; Only files
16942      <1>      ;mov     ax, 0810h ; Only directories

```



```

16943 000114A0 E8C574FFFF <1> call find_first_file
16944 <1> ;jnc short syschmod_2
16945 000114A5 720A <1> jc short syschmod_err
16946 <1>
16947 <1> ;; eax = 2 (File not found !)
16948 <1> ;cmp al, 2 ; ERR_NOT_FOUND
16949 <1> ;jne short syschmod_err
16950 <1>
16951 <1> ;and byte [Attributes], 10h
16952 <1> ;jz short syschmod_err
16953 <1>
16954 <1> ;; Directory not found !
16955 <1> ;mov al, 3 ; ERR_PATH_NOT_FOUND
16956 <1> ;jmp short syschmod_err
16957 <1>
16958 <1> syschmod_2:
16959 <1> and dx, dx ; Ambiguous filename chars used sign (DX>0)
16960 000114AA 7419 <1> jz short syschmod_3
16961 000114AC B81A000000 <1> mov eax, ERR_INV_FILE_NAME ; 'invalid file name !'
16962 <1> ;jmp short syschmod_err
16963 <1> syschmod_err:
16964 000114B1 A3[548E0100] <1> mov [u.r0], eax
16965 000114B6 A3[C08E0100] <1> mov [u.error], eax
16966 000114BB E8DA080000 <1> call reset_working_path
16967 000114C0 E999B5FFFF <1> jmp error
16968 <1> syschmod_3:
16969 <1> ; EDI = Directory buffer entry offset/address
16970 <1> ; BL = File (or Directory) Attributes
16971 <1> ; mov bl, [EDI+0Bh]
16972 <1>
16973 <1> ; check directory attributes
16974 000114C5 8A3D[1C810100] <1> mov bh, [Attributes] ; new attributes
16975 000114CB 80FF40 <1> cmp bh, 40h ;>=40 -> get file/directory attributes
16976 000114CE 7325 <1> jnb short syschmod_6
16977 <1>
16978 <1> ; set file/directory attributes
16979 000114D0 F6C307 <1> test bl, 7 ; system, hidden, readonly
16980 000114D3 7409 <1> jz short syschmod_4
16981 <1>
16982 <1> ; 19/01/2021
16983 000114D5 803D[A68E0100]00 <1> cmp byte [u.uid], 0 ; root (super user) ?
16984 000114DC 778E <1> ja short syschmod_perm_err
16985 <1> syschmod_4:
16986 <1> ; 19/12/2025
16987 <1> %if 0
16988 <1> cmp word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
16989 <1> je short syschmod_7
16990 <1> %endif
16991 000114DE 887F0B <1> mov [edi+0Bh], bh ; Attributes (New!)
16992 <1>
16993 000114E1 C605[897E0100]02 <1> mov byte [DirBuff_ValidData], 2 ; modified sign
16994 <1> ; to force write
16995 000114E8 E89498FFFF <1> call save_directory_buffer
16996 000114ED 72C2 <1> jc short syschmod_err
16997 <1>
16998 <1> syschmod_5:
16999 000114EF 8A1D[1C810100] <1> mov bl, [Attributes]
17000 <1> syschmod_6:
17001 000114F5 0FB6C3 <1> movzx eax, bl
17002 000114F8 A3[548E0100] <1> mov [u.r0], eax
17003 <1> ;mov dword [u.error], 0
17004 000114FD E97CB5FFFF <1> jmp sysret
17005 <1>
17006 <1> ; 19/12/2025
17007 <1> %if 0
17008 <1> syschmod_7:
17009 <1> sub eax, eax
17010 <1> mov ah, [DirBuff_Drv]
17011 <1> ; 26/09/2024 (BugFix)
17012 <1> sub ah, 'A'
17013 <1> mov esi, Logical_DOSDisks
17014 <1> add esi, eax
17015 <1> cmp byte [esi+LD_FSType], 0A1h
17016 <1> jnc short syschmod_8
17017 <1> mov al, ERR_INV_DATA ; 29 = Invalid Data
17018 <1> jmp short syschmod_err
17019 <1>
17020 <1> syschmod_8:
17021 <1> ; BH = New MS-DOS File Attributes
17022 <1> mov al, bh ; File/Directory Attributes
17023 <1> xor ah, ah ; Attributes in MS-DOS format sign
17024 <1> call change_fs_file_attributes
17025 <1> ;jc syschmod_err
17026 <1> ;jmp short syschmod_5
17027 <1> ; 23/07/2022
17028 <1> jnc short syschmod_5
17029 <1> jmp syschmod_err ; 26/09/2024
17030 <1> %endif
17031 <1>
17032 <1> sysdrive: ; Get/Set Current (Working) Drive (for user)
17033 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
17034 <1> ;
17035 <1> ; INPUT ->
17036 <1> ; BL = Logical DOS Drive number (0=A: ... 2=C:)
17037 <1> ; If BL = 0FFh -> Get Current Drive
17038 <1> ; OUTPUT ->
17039 <1> ; cf = 0 ->
17040 <1> ; AL = Current Drive number
17041 <1> ; AH = The Last Logical DOS Drive no.
17042 <1> ; cf = 1 -> Error code in AL
17043 <1> ;
17044 <1> ; Modified Registers: EAX (at the return of system call)
17045 <1> ;
17046 <1> ; NOTE: If the requested logical dos drive is ready,
17047 <1> ; it's current current directory will be the user's
17048 <1> ; (program's) current directory.
17049 <1> ; (When the program is terminated, MainProg -internal
17050 <1> ; shell- will reset the previous -current- logical drive
17051 <1> ; as current drive again).
17052 <1>
17053 00011502 80FBFF <1> cmp bl, 0FFh
17054 00011505 7435 <1> je short sysdrive_ok
17055 00011507 3A1D[7D2E0100] <1> cmp bl, [Last_DOS_DiskNo]
17056 0001150D 771E <1> ja short sysdrive_err
17057 <1>
17058 <1> ; Save current drive and reset mode
17059 <1> ; for 'reset_working_path' procedure (for MainProg)
17060 0001150F 30C0 <1> xor al, al
17061 00011511 66A3[5C830100] <1> mov [SWP_Mode], ax ; ah = 0
17062 00011517 A0[62770100] <1> mov al, [Current_Drv]
17063 0001151C FEC4 <1> inc ah ; mov ah, 1
17064 0001151E 66A3[5E830100] <1> mov [SWP_Drv], ax
17065 <1>
17066 00011524 88DA <1> mov dl, bl

```

```

17067 00011526 E8B561FFFF <1> call change_current_drive
17068 0001152B 730F <1> jnc short sysdrive_ok
17069 <1> sysdrive_err:
17070 0001152D C705[548E0100]0F00- <1> mov dword [u.r0], ERR_DRV_NOT_RDY ; 'drive not ready !'
17070 00011535 0000 <1>
17071 00011537 E922B5FFFF <1> jmp error
17072 <1> sysdrive_ok:
17073 0001153C A0[62770100] <1> mov al, [Current_Drv]
17074 00011541 8A25[7D2E0100] <1> mov ah, [Last_DOS_DiskNo]
17075 00011547 A3[548E0100] <1> mov [u.r0], eax
17076 0001154C E92DB5FFFF <1> jmp sysret
17077 <1>
17078 <1> sysdir: ; Get Current (Working) Drive & Directory (for user)
17079 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
17080 <1> ;
17081 <1> ; INPUT ->
17082 <1> ; EBX = Current directory name buffer address
17083 <1> ; (Buffer length = 92 bytes)
17084 <1> ; OUTPUT ->
17085 <1> ; AL = Current drive (0=A: .. 2=C:)
17086 <1> ; If CF = 1 -> AL = error code
17087 <1> ;
17088 <1> ; Modified Registers: EAX (at the return of system call)
17089 <1> ;
17090 <1> ; Note: Required directory name buffer length may be
17091 <1> ; <= 92 bytes for current TRDOS 386 version.
17092 <1> ; (7*12 name chars + 7 slash + 0)
17093 <1>
17094 00011551 89E5 <1> mov ebp, esp
17095 00011553 83EC60 <1> sub esp, 96
17096 00011556 53 <1> push ebx ; User's buffer address
17097 00011557 30D2 <1> xor dl, dl ; 0 = current drive
17098 00011559 E8238EFFFF <1> call get_current_directory
17099 0001155E 72CD <1> jc short sysdrive_err ; 'drive not ready !' error
17100 00011560 89E6 <1> mov esi, esp ; System's buffer address
17101 00011562 5F <1> pop edi ; User's buffer address
17102 <1> ; ecx = transfer (byte) count (<=92)
17103 00011563 E8A3F5FFFF <1> call transfer_to_user_buffer
17104 00011568 89EC <1> mov esp, ebp
17105 0001156A 730F <1> jnc short sysdir_ok
17106 <1> sysdir_err:
17107 0001156C C705[548E0100]2E00- <1> mov dword [u.r0], ERR_BUFFER ; 'buffer error !'
17107 00011574 0000 <1>
17108 00011576 E9E3B4FFFF <1> jmp error
17109 <1> sysdir_ok:
17110 0001157B 8A0D[62770100] <1> mov cl, [Current_Drv]
17111 00011581 890D[548E0100] <1> mov [u.r0], ecx
17112 00011587 E9F2B4FFFF <1> jmp sysret
17113 <1>
17114 <1> sysldrvt: ; Get copy of Logical DOS Drive Description Table
17115 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
17116 <1> ;
17117 <1> ; INPUT ->
17118 <1> ; BL = Logical DOS drive number (zero based)
17119 <1> ; ECX = Logical DOS drv desc table buffer addr
17120 <1> ; (Buffer length = 256 bytes)
17121 <1> ; OUTPUT ->
17122 <1> ; cf = 0 ->
17123 <1> ; AL = Current Drive number
17124 <1> ; AH = The Last Logical DOS Drive no.
17125 <1> ; cf = 1 -> Error code in AL
17126 <1> ; AH = The Last Logical DOS Drive no.
17127 <1> ;
17128 <1> ; Modified Registers: EAX (at the return of system call)
17129 <1> ;
17130 <1> ; Note: Required description table buffer length is
17131 <1> ; 256 bytes for current TRDOS 386 version.
17132 <1>
17133 0001158C 89CF <1> mov edi, ecx ; Destination address (user space)
17134 0001158E 88DC <1> mov ah, bl
17135 00011590 30C0 <1> xor al, al
17136 00011592 BE00010900 <1> mov esi, Logical_DOSDisks
17137 00011597 01C6 <1> add esi, eax ; Source address (system space)
17138 00011599 B900010000 <1> mov ecx, 256 ; Byte count
17139 <1> ; Logical Dos Drv Desc Table size
17140 0001159E E868F5FFFF <1> call transfer_to_user_buffer
17141 000115A3 72C7 <1> jc short sysdir_err
17142 000115A5 8A2D[7D2E0100] <1> mov ch, [Last_DOS_DiskNo]
17143 000115AB EBCE <1> jmp short sysdir_ok
17144 <1>
17145 <1> systime: ; Get System Date&Time
17146 <1> ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
17147 <1> ;
17148 <1> ; INPUT -> BL =
17149 <1> ; 0 = Get Date&Time in Unix/Epoch format
17150 <1> ; 1 = Get Time in MSDOS format
17151 <1> ; 2 = Get Date in MSDOS format
17152 <1> ; 3 = Get Date&Time in MSDOS format
17153 <1> ; 4 & other values =
17154 <1> ; System timer ticks will be returned
17155 <1> ; in EAX and Carry Flag will be set.
17156 <1> ; (CF will not be set if BL = 4)
17157 <1> ; OUTPUT ->
17158 <1> ; For BL input = 3
17159 <1> ; EAX = Current Time (RTC)
17160 <1> ; AL = Second (DL in MSDOS)
17161 <1> ; AH = Minute (CL in MSDOS)
17162 <1> ; HW of EAX = Hour (CH in MSDOS)
17163 <1> ; EDX = Current System Date (RTC)
17164 <1> ; DL = Day (DL in MSDOS)
17165 <1> ; DH = Month (DH in MSDOS)
17166 <1> ; HW of EDX = Year (CX in MSDOS)
17167 <1> ;
17168 <1> ; For BL input = 2
17169 <1> ; EAX = Current System Date (RTC)
17170 <1> ; DL = Day (DL in MSDOS)
17171 <1> ; DH = Month (DH in MSDOS)
17172 <1> ; HW of EDX = Year (CX in MSDOS)
17173 <1> ;
17174 <1> ; For BL input = 1
17175 <1> ; EAX = Current Time (RTC)
17176 <1> ; AL = Second (DL in MSDOS)
17177 <1> ; AH = Minute (CL in MSDOS)
17178 <1> ; HW of EAX = Hour (CH in MSDOS)
17179 <1> ;
17180 <1> ; For BL input = 0
17181 <1> ; EAX = Unix (Epoch) Time Ticks/Seconds
17182 <1> ;
17183 <1> ; For BL input = 4
17184 <1> ; EAX = System timer ticks
17185 <1> ;
17186 <1> ; If CF = 1 (for other values of BL input)
17187 <1> ; EAX = System timer ticks (no error code!)
17188 <1> ;

```

```

17189      <1>      ; Modified Registers: EAX, (EDX)
17190      <1>      ;      (at the return of system call)
17191      <1>      ;
17192      <1>      ;
17193      000115AD 20DB      <1>      and     b1, b1
17194      000115AF 750F      <1>      jnz     short systime_1
17195      000115B1 E81B59FFFF <1>      call    epoch
17196      <1>      systime_0:
17197      000115B6 A3[548E0100] <1>      mov     [u.r0], eax
17198      000115BB E9BEB4FFFF <1>      jmp     sysret
17199      <1>      systime_1:
17200      000115C0 80FB04      <1>      cmp     b1, 4
17201      000115C3 7211      <1>      jb      short systime_2
17202      000115C5 A1[20770100] <1>      mov     eax, [TIMER_LH] ; 18.2 Hz timer ticks
17203      <1>      ; Note: [TIMER_LH] may be set
17204      <1>      ; to wrong timer value due to
17205      <1>      ; program functions.
17206      <1>      ; (This value must not be
17207      <1>      ; accepted as [TIMER_LH]/18.2
17208      <1>      ; seconds since the midnight.)
17209      000115CA 76EA      <1>      jna     short systime_0
17210      000115CC A3[548E0100] <1>      mov     [u.r0], eax
17211      000115D1 E988B4FFFF <1>      jmp     error ; cf = 1 & [u.r0] = eax = timer ticks
17212      <1>      ;
17213      <1>      systime_2:
17214      <1>      ;push    ebx
17215      000115D6 E87058FFFF <1>      call    get_rtc_date_time
17216      <1>      ;pop     ebx
17217      000115DB F6C301      <1>      test    b1, 1
17218      000115DE 7429      <1>      jz      short systime_4
17219      000115E0 30E4      <1>      xor     ah, ah
17220      000115E2 A0[6A730100] <1>      mov     al, [hour]
17221      000115E7 88C2      <1>      mov     dl, al
17222      000115E9 C1E010      <1>      shl     eax, 16
17223      000115EC A0[6E730100] <1>      mov     al, [second]
17224      000115F1 8A25[6C730100] <1>      mov     ah, [minute]
17225      000115F7 F6C302      <1>      test    b1, 2
17226      000115FA 74BA      <1>      jz      short systime_0
17227      <1>      ; Check time & date match risk
17228      <1>      ; (23:59:59 may cause to wrong
17229      <1>      ; date -new day with previous date-...)
17230      000115FC 80FA17      <1>      cmp     dl, 23
17231      000115FF 7206      <1>      jb      short systime_3
17232      00011601 663D3B3B <1>      cmp     ax, (59*256)+59 ; if hour is 23:59:59
17233      00011605 73CF      <1>      jnb     short systime_2 ; wait for 1 second
17234      <1>      systime_3:
17235      <1>      ; eax = time
17236      00011607 89C6      <1>      mov     esi, eax
17237      <1>      systime_4:
17238      00011609 66A1[64730100] <1>      mov     ax, [year]
17239      0001160F C1E010      <1>      shl     eax, 16
17240      00011612 A0[68730100] <1>      mov     al, [day]
17241      00011617 8A25[66730100] <1>      mov     ah, [month]
17242      <1>      ; eax = date
17243      0001161D 80E301      <1>      and     b1, 1
17244      00011620 7494      <1>      jz      short systime_0
17245      00011622 96      <1>      xchg     esi, eax
17246      <1>      ; eax = time, esi = date
17247      00011623 8B2D[508E0100] <1>      mov     ebp, [u.usp] ; EBP points to user's registers
17248      <1>      ; (user) edx <-- (system) esi
17249      00011629 897514      <1>      mov     [ebp+20], esi ; return to user with EDX value
17250      0001162C EB88      <1>      jmp     short systime_0
17251      <1>      ;
17252      <1>      sysstime: ; Set System Date&Time
17253      <1>      ; 31/12/2017
17254      <1>      ; 30/12/2017 (TRDOS 386 = TRDOS v2.0)
17255      <1>      ;
17256      <1>      ; INPUT -> BL =
17257      <1>      ;      0 = Set Date&Time in Unix/Epoch format
17258      <1>      ;      1 = Set Time in MSDOS format
17259      <1>      ;      2 = Set Date in MSDOS format
17260      <1>      ;      3 = Set Date&Time in MSDOS format
17261      <1>      ;      4 = Set System Timer (Ticks)
17262      <1>      ;      5 = Convert/Save current time to/as
17263      <1>      ;      18.2 Hz system timer ticks
17264      <1>      ;      6 = Convert MSDOS Date&Time to UNIX format
17265      <1>      ;      without setting system date&time ; (test)
17266      <1>      ;      7 = Convert UNIX Date&Time to MSDOS format
17267      <1>      ;      without setting system date&time ; (test)
17268      <1>      ;      8-0FFh = invalid !
17269      <1>      ;      ECX = Time (or Timer) value in selected format
17270      <1>      ;      EDX = Date value in MSDOS format if BL=2,3,6
17271      <1>      ;
17272      <1>      ; OUTPUT ->
17273      <1>      ;      If CF = 0 ->
17274      <1>      ;      EAX = Set value
17275      <1>      ;      If CF = 1 -> (invalid BL input)
17276      <1>      ;      EAX = Ticks count [TIMER_LH]
17277      <1>      ;
17278      <1>      ;
17279      0001162E 20DB      <1>      and     b1, b1 ; 0
17280      00011630 7511      <1>      jnz     short sysstime_0
17281      00011632 89C8      <1>      mov     eax, ecx
17282      00011634 E81D59FFFF <1>      call    convert_from_epoch
17283      00011639 E8BD59FFFF <1>      call    set_rtc_date_time
17284      0001163E E93BB4FFFF <1>      jmp     sysret
17285      <1>      sysstime_0:
17286      00011643 80FB08      <1>      cmp     b1, 8
17287      00011646 722D      <1>      jb      short sysstime_1
17288      <1>      ; invalid input (>7)
17289      00011648 A1[20770100] <1>      mov     eax, [TIMER_LH] ; 18.2 Hz timer ticks
17290      <1>      ; Note: [TIMER_LH] may be set
17291      <1>      ; to wrong timer value due to
17292      <1>      ; program functions.
17293      <1>      ; (This value must not be
17294      <1>      ; accepted as [TIMER_LH]/18.2
17295      <1>      ; seconds since the midnight.)
17296      0001164D A3[548E0100] <1>      mov     [u.r0], eax
17297      00011652 E907B4FFFF <1>      jmp     error ; cf = 1 & [u.r0] = eax = timer ticks
17298      <1>      ;
17299      <1>      sysstime_8:
17300      <1>      ; BL = 7
17301      00011657 89C8      <1>      mov     eax, ecx ; seconds since 1/1/1970 00:00:00
17302      00011659 E8F858FFFF <1>      call    convert_from_epoch
17303      0001165E 30E4      <1>      xor     ah, ah
17304      00011660 A0[6A730100] <1>      mov     al, [hour]
17305      00011665 C1E010      <1>      shl     eax, 16
17306      00011668 A0[6E730100] <1>      mov     al, [second]
17307      0001166D 8A25[6C730100] <1>      mov     ah, [minute]
17308      00011673 EB92      <1>      jmp     short systime_3
17309      <1>      ;
17310      <1>      sysstime_1:
17311      00011675 80FB04      <1>      cmp     b1, 4
17312      00011678 743F      <1>      je      short sysstime_2 ; set system timer ticks

```

```

17313 0001167A 80FB05 <1> cmp bl, 5
17314 0001167D 754B <1> jne short sysstime_4
17315 <1> ; convert current time to system timer ticks (18.2Hz)
17316 0001167F E8C757FFFF <1> call get_rtc_date_time
17317 00011684 0FB60D[6A730100] <1> movzx ecx, byte [hour]
17318 00011688 B8100E0000 <1> mov eax, 60*60 ; 1 hour = 3600 seconds
17319 00011690 F7E1 <1> mul ecx
17320 00011692 89C3 <1> mov ebx, eax
17321 00011694 B13C <1> mov cl, 60 ; 1 minute = 60 seconds
17322 00011696 0FB605[6C730100] <1> movzx eax, byte [minute]
17323 0001169D F7E1 <1> mul ecx
17324 0001169F 01D8 <1> add eax, ebx
17325 000116A1 8A0D[6E730100] <1> mov cl, [second]
17326 000116A7 01C8 <1> add eax, ecx
17327 000116A9 B1B6 <1> mov cl, 182
17328 000116AB F7E1 <1> mul ecx
17329 000116AD 83C009 <1> add eax, 9
17330 000116B0 83D200 <1> adc edx, 0
17331 000116B3 B10A <1> mov cl, 10
17332 000116B5 F7F1 <1> div ecx
17333 <1> ; eax = ((182*seconds)+9)/10
17334 000116B7 89C1 <1> mov ecx, eax
17335 <1> sysstime_2:
17336 000116B9 890D[20770100] <1> mov [TIMER_LH], ecx ; 18.2 * seconds
17337 <1> sysstime_3:
17338 000116BF 890D[548E0100] <1> mov [u.r0], ecx
17339 000116C5 E9B4B3FFFF <1> jmp sysret
17340 <1> sysstime_4:
17341 000116CA 80FB06 <1> cmp bl, 6
17342 000116CD 7788 <1> ja short sysstime_8
17343 <1>
17344 000116CF 890D[548E0100] <1> mov [u.r0], ecx
17345 <1>
17346 000116D5 880D[6E730100] <1> mov [second], cl
17347 000116DB 882D[6C730100] <1> mov [minute], ch
17348 000116E1 C1E910 <1> shr ecx, 16
17349 000116E4 880D[6A730100] <1> mov [hour], cl
17350 <1> ; BL = 1,2,3,6
17351 000116EA 80FB01 <1> cmp bl, 1
17352 000116ED 762A <1> jna short sysstime_5
17353 <1> ; BL = 2,3,6
17354 000116EF 8815[68730100] <1> mov [day], dl
17355 000116F5 8835[66730100] <1> mov [month], dh
17356 000116FB C1EA10 <1> shr edx, 16
17357 000116FE 668915[64730100] <1> mov [year], dx
17358 00011705 80E303 <1> and bl, 3
17359 00011708 742D <1> jz short sysstime_7 ; 6
17360 <1> ; BL = 2,3
17361 0001170A F6C301 <1> test bl, 1
17362 0001170D 7419 <1> jz short sysstime_6 ; 2
17363 <1> ; BL = 3
17364 0001170F E8E758FFFF <1> call set_rtc_date_time
17365 00011714 E965B3FFFF <1> jmp sysret
17366 <1> sysstime_5:
17367 <1> ; BL = 1
17368 00011719 E81E59FFFF <1> call set_time_bcd
17369 0001171E E85C4CFFFF <1> call set_rtc_time
17370 00011723 E956B3FFFF <1> jmp sysret
17371 <1> sysstime_6:
17372 <1> ; BL = 2
17373 00011728 E8E258FFFF <1> call set_date_bcd
17374 0001172D E8B04CFFFF <1> call set_rtc_date
17375 00011732 E947B3FFFF <1> jmp sysret
17376 <1> sysstime_7:
17377 <1> ; BL = 6
17378 <1> ; [year], [month], [day],
17379 <1> ; [hour], [minute], [second]
17380 00011737 E89A57FFFF <1> call convert_to_epoch
17381 0001173C 89C1 <1> mov ecx, eax ; seconds since 1/1/1970 00:00:00
17382 0001173E E97CFFFFF <1> jmp sysstime_3
17383 <1>
17384 <1> sysrename: ; Rename File (or Directory)
17385 <1> ; 19/12/2025 - TRDOS 386 v2.0.10
17386 <1> ; 08/08/2022
17387 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
17388 <1> ; 19/01/2021
17389 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
17390 <1> ;
17391 <1> ; INPUT ->
17392 <1> ;
17393 <1> ; EBX = File/Directory (ASCIIIZ) name address
17394 <1> ; ECX = New name (in same dir, no path name)
17395 <1> ; OUTPUT ->
17396 <1> ; cf = 0 -> EAX = 0
17397 <1> ; cf = 1 -> Error code in AL
17398 <1> ; 19/01/2021
17399 00011743 803D[A68E0100]00 <1> cmp byte [u.uid], 0 ; root (super user) ?
17400 <1> ; jna short sysrename_0
17401 <1> ; 23/07/2022
17402 0001174A 7717 <1> ja short sysrename_perm_err
17403 <1>
17404 <1> sysrename_perm_err:
17405 <1> ; mov dword [u.r0], ERR_PERM_DENIED
17406 <1> ; mov eax, ERR_PERM_DENIED ; 'permission denied !'
17407 <1> ; mov [u.r0], eax
17408 <1> ; mov [u.error], eax
17409 <1> ; jmp error
17410 <1>
17411 <1> sysrename_0:
17412 0001174C 51 <1> push ecx ; new file name address (in user space)
17413 0001174D 89DE <1> mov esi, ebx
17414 <1> ; file name is forced, change directory as temporary
17415 <1> ; mov ax, 1
17416 <1> ; mov [FFF_Valid], ah ; 0 ; reset
17417 <1> ; call set_working_path
17418 0001174F E871050000 <1> call set_working_path_x
17419 00011754 7321 <1> jnc short sysrename_1
17420 00011756 21C0 <1> and eax, eax ; 0 -> Bad Path!
17421 00011758 753B <1> jnz short sysrename_err
17422 <1> ; eax = 0
17423 <1> sysrename_path_not_found:
17424 0001175A B813000000 <1> mov eax, ERR_INV_PATH_NAME ; 'Bad path name !'
17425 <1> ; 23/07/2022
17426 0001175F B013 <1> mov al, ERR_INV_PATH_NAME
17427 00011761 EB32 <1> jmp short sysrename_err
17428 <1> sysrename_err:
17429 <1> ; pop ecx ; new file name address (in user space)
17430 <1> sysrename_error:
17431 <1> ; mov [u.r0], eax
17432 <1> ; mov [u.error], eax
17433 <1> ; call reset_working_path
17434 <1> ; jmp error
17435 <1>
17436 <1> ; 23/07/2022

```

```

17437 <1> sysrename_perm_err:
17438 <1> ;mov     dword [u.r0], ERR_PERM_DENIED
17439 <1> mov     eax, ERR_PERM_DENIED ; 'permission denied !'
17440 <1> mov     [u.r0], eax
17441 <1> mov     [u.error], eax
17442 <1> jmp     error
17443 <1>
17444 <1> sysrename_1:
17445 <1> mov     al, 08h ; Except volume labels (& long names)
17446 <1> mov     al, [Attributes]
17447 <1> and     al, 10h ;
17448 <1> ;mov     esi, FindFile_Name
17449 <1> ;mov     ax, 1800h ; Only files
17450 <1> ;mov     ax, 0810h ; Only directories
17451 <1> mov     ax, 0800h ; Find File or Directory
17452 <1> call    find_first_file
17453 <1> ;jnc     short sysrename_2
17454 <1> jc      short sysrename_err
17455 <1> sysrename_2:
17456 <1> ; ESI = Directory Entry (FindFile_DirEntry) Location
17457 <1> ; EDI = Directory Buffer Directory Entry Location
17458 <1> ; EAX = File Size
17459 <1> ; BL = Attributes of The File/Directory
17460 <1> ; BH = Long Name Yes/No Status (>0 is YES)
17461 <1> ; DX > 0 : Ambiguous filename chars are used
17462 <1>
17463 <1> and     dx, dx ; Ambiguous filename chars used sign (DX>0)
17464 <1> jz      short sysrename_3
17465 <1> mov     eax, ERR_INV_FILE_NAME ; 'invalid file name !'
17466 <1> jmp     short sysrename_err
17467 <1> ; 23/07/2022
17468 <1> sysrename_err:
17469 <1> pop     ecx ; new file name address (in user space)
17470 <1> sysrename_error:
17471 <1> mov     [u.r0], eax
17472 <1> mov     [u.error], eax
17473 <1> call    reset_working_path
17474 <1> jmp     error
17475 <1> sysrename_3:
17476 <1> ; EDI = Directory buffer entry offset/address
17477 <1> ; BL = File (or Directory) Attributes
17478 <1> ; mov     bl, [EDI+0Bh]
17479 <1>
17480 <1> pop     edx ; new file name address (in user space)
17481 <1>
17482 <1> ; check file/directory attributes
17483 <1> test    bl, 7 ; system, hidden, readonly
17484 <1> jnz     short sysrename_perm_err
17485 <1> sysrename_4:
17486 <1> ; 19/12/2025
17487 <1> ;cmp     word [edi+DirEntry_NTRes], 01A1h ; Singlix FS
17488 <1> ;je      short sysrename_perm_err ; -temporary!-
17489 <1>
17490 <1> ; save old file name & file info (FFF structure)
17491 <1> mov     esi, FindFile_Drv
17492 <1> mov     edi, SourceFile_Drv
17493 <1> ;mov     ecx, 128/4
17494 <1> ; 23/07/2022
17495 <1> sub     ecx, ecx
17496 <1> mov     cl, 128/4
17497 <1> rep     movsd
17498 <1>
17499 <1> mov     esi, edx ; new file name address (in user space)
17500 <1> mov     edi, DestinationFile_Drv
17501 <1> call    parse_path_name
17502 <1> jc      short sysrename_error ; eax = 1 (Bad file name)
17503 <1>
17504 <1> ; same drive ?
17505 <1> mov     al, [FindFile_Drv]
17506 <1> cmp     al, [DestinationFile_Drv]
17507 <1> ;jne     short sysrename_perm_err ; Permission denied
17508 <1> jne     short sysrename_5 ; Bad file name
17509 <1>
17510 <1> ; no path name !? (rename file in same directory)
17511 <1> cmp     byte [DestinationFile_Directory], 20h
17512 <1> jna     short sysrename_6
17513 <1> sysrename_5:
17514 <1> mov     eax, ERR_BAD_CMD_ARG ; 1 = Bad file name
17515 <1> ; (Bad argument)
17516 <1> jmp     short sysrename_error
17517 <1> sysrename_6:
17518 <1> cmp     byte [DestinationFile_Name], 20h
17519 <1> jna     short sysrename_5
17520 <1>
17521 <1> mov     esi, DestinationFile_Name
17522 <1> call    check_filename ; is it a valid msdos file name?
17523 <1> jc      short sysrename_error ; 26 = ERR_INV_FILE_NAME
17524 <1>
17525 <1> ;mov     esi, DestinationFile_Name
17526 <1> mov     ax, 0800h ; Find File or Directory
17527 <1> call    find_first_file
17528 <1> jc      short sysrename_7
17529 <1>
17530 <1> mov     eax, ERR_FILE_EXISTS ; file already exists !
17531 <1> jmp_sysrename_err: ; 08/08/2022
17532 <1> jmp     sysrename_error
17533 <1> sysrename_7:
17534 <1> ; eax = 2 (File not found !)
17535 <1> cmp     al, 2 ; ERR_NOT_FOUND
17536 <1> ;jne     short sysrename_error
17537 <1> ; 08/08/2022
17538 <1> jne     short jmp_sysrename_err
17539 <1>
17540 <1> ; 31/12/2017
17541 <1> ; Following code is also part of 'rename_file' in
17542 <1> ; 'trdosk3.s' (MainProg's 'rename' command) ; 13/11/2017
17543 <1> mov     esi, DestinationFile_Name ; (Rename_NewName)
17544 <1> mov     cx, [SourceFile_DirEntryNumber]
17545 <1> mov     ax, [SourceFile_DirEntry+20] ; First cluster, HW
17546 <1> shl     eax, 16
17547 <1> mov     ax, [SourceFile_DirEntry+26] ; First cluster, LW
17548 <1> movzx   ebx, byte [SourceFile_LongNameEntryLength]
17549 <1> call    rename_directory_entry
17550 <1> ;jc      short sysrename_error
17551 <1> ; 08/08/2022
17552 <1> jc      short jmp_sysrename_err
17553 <1> ;xor     eax, eax
17554 <1> mov     [u.r0], eax ; 0
17555 <1> ;mov     [u.error], eax
17556 <1> call    reset_working_path
17557 <1> jmp     sysret
17558 <1>
17559 <1> system: ; Get Total&Free Memory amount
17560 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)

```

```

17561 <1> ;
17562 <1> ; INPUT ->
17563 <1> ; none
17564 <1> ; OUTPUT ->
17565 <1> ; EAX = Total memory count (in bytes)
17566 <1> ; EBX = Virtually available memory amount (in bytes)
17567 <1> ; = 4GB - CORE (4MB)
17568 <1> ; ECX = Free memory count (in bytes)
17569 <1> ; EDX = Calculated free memory count (in bytes)
17570 <1>
17571 0001184E A1[A4760100] <1> mov eax, [memory_size] ; in pages
17572 00011853 C1E00C <1> shl eax, 12 ; in bytes
17573 00011856 A3[548E0100] <1> mov [u.r0], eax
17574 0001185B E8D729FFFF <1> call calc_free_mem
17575 <1> ; edx = calculated free pages
17576 <1> ; ecx = 0
17577 00011860 8B2D[508E0100] <1> mov ebp, [u.usp] ; EBP points to user's registers
17578 00011866 C745100000C0FF <1> mov dword [ebp+16], ECORE ; EBX (for user)
17579 <1> ; 0FFC00000h ; 4GB - 4MB
17580 0001186D C1E20C <1> shl edx, 12
17581 00011870 895514 <1> mov [ebp+20], edx ; EDX (for user)
17582 00011873 8B0D[A8760100] <1> mov ecx, [free_pages]
17583 00011879 C1E10C <1> shl ecx, 12 ; free bytes
17584 0001187C 894D18 <1> mov [ebp+24], ecx ; ECX (for user)
17585 <1> ;mov [free_pages], edx
17586 0001187F E9FAB1FFFF <1> jmp sysret
17587 <1>
17588 <1> sysprompt:
17589 <1> ; Set TRDOS 386 Command Interpreter (MainProg) prompt
17590 <1> ; 30/07/2022 (TRDOS 386 Kernel v2.0.5)
17591 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
17592 <1> ;
17593 <1> ; INPUT ->
17594 <1> ; EBX = 0 -> use default prompt
17595 <1> ; EBX > 0 -> prompt string (ASCIIZ) address
17596 <1> ; (Max. 11 characters except ZERO tail)
17597 <1> ; OUTPUT ->
17598 <1> ; (EAX = 0)
17599 <1> ; CF = 0 -> Successful
17600 <1> ; CF = 1 -> Failed
17601 <1>
17602 00011884 21DB <1> and ebx, ebx
17603 00011886 750A <1> jnz short sysprompt_0
17604 <1>
17605 00011888 E8C46AFFFF <1> call default_command_prompt ; '['+TRDOS'+']'
17606 0001188D E9ECB1FFFF <1> jmp sysret
17607 <1>
17608 <1> sysprompt_0:
17609 00011892 31C0 <1> xor eax, eax
17610 00011894 A3[548E0100] <1> mov [u.r0], eax
17611 00011899 89DE <1> mov esi, ebx
17612 <1> ;mov ecx, 12
17613 <1> ; 30/07/2022
17614 0001189B 31C9 <1> xor ecx, ecx
17615 0001189D B10C <1> mov cl, 12
17616 0001189F 89E5 <1> mov ebp, esp
17617 000118A1 29CC <1> sub esp, ecx
17618 000118A3 49 <1> dec ecx ; 11
17619 000118A4 89E7 <1> mov edi, esp
17620 000118A6 E8AAF2FFFF <1> call transfer_from_user_buffer
17621 000118AB 7211 <1> jc short sysprompt_err
17622 000118AD 803E20 <1> cmp byte [esi], 20h
17623 000118B0 760C <1> jna short sysprompt_err
17624 000118B2 E8AC6AFFFF <1> call set_command_prompt
17625 000118B7 89EC <1> mov esp, ebp
17626 000118B9 E9C0B1FFFF <1> jmp sysret
17627 <1> sysprompt_err:
17628 <1> syspath_err:
17629 000118BE 89EC <1> mov esp, ebp
17630 000118C0 E999B1FFFF <1> jmp error
17631 <1>
17632 <1> syspath:
17633 <1> ; Get/Set Run Path
17634 <1> ;
17635 <1> ; 30/07/2022 (TRDOS 386 Kernel v2.0.5)
17636 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
17637 <1> ;
17638 <1> ; INPUT ->
17639 <1> ; EBX = 0 -> get path (to buffer address in ECX)
17640 <1> ; EBX > 0 -> set path
17641 <1> ; EBX = Path string buffer address (ASCIIZ)
17642 <1> ; (Path description except 'PATH=')
17643 <1> ; ECX = Buffer address (if EBX = 0)
17644 <1> ; (ECX will not be used if EBX > 0)
17645 <1> ; DL = Buffer size (0 = 256 byte)
17646 <1> ;
17647 <1> ; OUTPUT ->
17648 <1> ; CF = 0 -> Successful (EAX = String length)
17649 <1> ; CF = 1 -> Failed (EAX = 0)
17650 <1> ;
17651 <1> ; NOTE: 'PATH=' or 'PATH' must be excluded
17652 <1> ; (It must not be at the beginning of the string.)
17653 <1>
17654 000118C5 89E5 <1> mov ebp, esp
17655 000118C7 81EC00010000 <1> sub esp, 256
17656 000118CD 89E7 <1> mov edi, esp
17657 <1>
17658 000118CF 31C0 <1> xor eax, eax
17659 000118D1 A3[548E0100] <1> mov [u.r0], eax
17660 <1>
17661 000118D6 21DB <1> and ebx, ebx
17662 000118D8 752A <1> jnz short syspath_0
17663 <1>
17664 <1> ; EBX = 0 -> get run path
17665 000118DA 89CB <1> mov ebx, ecx ; buffer addr (in user's mem space)
17666 000118DC BE[532F0100] <1> mov esi, Cmd_Path ; 'PATH' address
17667 000118E1 0FB6CA <1> movzx ecx, dl
17668 <1> ;sub cl, 1 ; 0 -> 255, 1 -> 0
17669 <1> ;adc cx, 1 ; 255 -> 256, 0 -> 1
17670 <1> ; 30/07/2022
17671 000118E4 FEC9 <1> dec cl ; 0 -> 255, 1 -> 0
17672 000118E6 41 <1> inc ecx ; 255 -> 256, 0 -> 1
17673 <1> ; EDI = Output buffer
17674 <1> ; CX = Buffer length
17675 <1> ; AL = 0 -> use ASCIIZ word in [ESI]
17676 <1> ; ESI = 'PATH' address (with zero tail)
17677 000118E7 E8BB81FFFF <1> call get_environment_string
17678 000118EC 72D0 <1> jc short syspath_err
17679 000118EE 89DF <1> mov edi, ebx ; User's buffer address
17680 000118F0 89E6 <1> mov esi, esp
17681 <1> ; EDI = User's buffer address
17682 <1> ; ECX = transfer (byte) count
17683 000118F2 E814F2FFFF <1> call transfer_to_user_buffer
17684 000118F7 72C5 <1> jc short syspath_err

```

```

17685 000118F9 890D[548E0100] <1> mov [u.r0], ecx
17686 000118FF E97AB1FFFF <1> jmp sysret
17687 <1>
17688 <1> syspath_0:
17689 00011904 89DE <1> mov esi, ebx
17690 00011906 0FB6CA <1> movzx ecx, dl
17691 <1> ;sub cl, 1 ; 0 -> 255, 1 -> 0
17692 <1> ;adc cx, 1 ; 255 -> 256, 0 -> 1
17693 <1> ; 30/07/2022
17694 00011909 FEC9 <1> dec cl ; 0 -> 255, 1 -> 0
17695 0001190B 41 <1> inc ecx ; 255 -> 256, 0 -> 1
17696 0001190C E844F2FFFF <1> call transfer_from_user_buffer
17697 00011911 72AB <1> jc short syspath_err
17698 <1> ;(*) 'PATH=' will be added to
17699 <1> ; the head of the string
17700 00011913 83EC08 <1> sub esp, 8 ;(*)
17701 00011916 89FE <1> mov esi, edi ;(*)
17702 00011918 E86E81FFFF <1> call set_path_x ;(*)
17703 0001191D 729F <1> jc short syspath_err
17704 0001191F 8915[548E0100] <1> mov [u.r0], edx ; run path string length
17705 00011925 E954B1FFFF <1> jmp sysret
17706 <1>
17707 <1> sysenv:
17708 <1> ; Get/Set Environment Variables
17709 <1>
17710 <1> ; 30/07/2022
17711 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
17712 <1> ; 31/12/2017 (TRDOS 386 = TRDOS v2.0)
17713 <1>
17714 <1> ; INPUT ->
17715 <1> ; EBX = 0 -> get (all) environment variables
17716 <1> ; (Required Buffer length = 512 bytes)
17717 <1> ; EBX > 0 -> set (one) environment variable
17718 <1> ; (If there is not a '=' after
17719 <1> ; the environment variable name, it will
17720 <1> ; accepted as 'get environment variable'.)
17721 <1> ; EBX = Buffer address
17722 <1> ; ECX = Buffer address (if EBX = 0)
17723 <1> ; (ECX will not be used if EBX > 0)
17724 <1> ; (Note: Buffer size is 512 bytes.)
17725 <1> ; DL = Buffer size (0 = 256 byte)
17726 <1> ; (For one environment variable)
17727 <1>
17728 <1> ; OUTPUT ->
17729 <1> ; (EAX = 0)
17730 <1> ; CF = 0 -> Successful (EAX = String length)
17731 <1> ; CF = 1 -> Failed (EAX = 0)
17732 <1>
17733 <1> ; Note: Environment variable name, for example,
17734 <1> ; 'PATH=' must be included at the beginning
17735 <1> ; of the environment string. If the variable
17736 <1> ; name is as 'PATH' but it is not as 'PATH='
17737 <1> ; the variable string (row) will be returned.
17738 <1> ; If variable name is as 'PATH=' but there is
17739 <1> ; not a following text after the variable name,
17740 <1> ; the environment variable will be reset/deleted.
17741 <1>
17742 0001192A 89E5 <1> mov ebp, esp
17743 0001192C 81EC00020000 <1> sub esp, 512
17744 00011932 89E7 <1> mov edi, esp
17745 <1>
17746 00011934 31C0 <1> xor eax, eax
17747 00011936 A3[548E0100] <1> mov [u.r0], eax
17748 <1>
17749 0001193B 21DB <1> and ebx, ebx
17750 0001193D 751F <1> jnz short sysenv_0
17751 <1>
17752 <1> ; EBX = 0 -> get (all) environment variables
17753 0001193F 89EC <1> mov esp, ebp
17754 00011941 BE00300900 <1> mov esi, Env_Page ; Environment page
17755 00011946 89CF <1> mov edi, ecx ; buffer addr (in user's mem space)
17756 <1> ;mov ecx, 512
17757 <1> ; 30/07/2022
17758 00011948 29C9 <1> sub ecx, ecx
17759 0001194A B502 <1> mov ch, 2
17760 <1> ; ecx = 512
17761 0001194C E8BAF1FFFF <1> call transfer_to_user_buffer
17762 <1> ;jc error
17763 <1> ; 23/07/2022
17764 00011951 725B <1> jc short sysenv_error
17765 00011953 890D[548E0100] <1> mov [u.r0], ecx
17766 00011959 E920B1FFFF <1> jmp sysret
17767 <1>
17768 <1> sysenv_0:
17769 0001195E 89DE <1> mov esi, ebx ; * ; user's buffer address
17770 00011960 0FB6CA <1> movzx ecx, dl
17771 <1> ;sub cl, 1 ; 0 -> 255, 1 -> 0
17772 <1> ;adc cx, 1 ; 255 -> 256, 0 -> 1
17773 <1> ; 30/07/2022
17774 00011963 FEC9 <1> dec cl ; 0 -> 255, 1 -> 0
17775 00011965 41 <1> inc ecx ; 255 -> 256, 0 -> 1
17776 00011966 E8EAF1FFFF <1> call transfer_from_user_buffer
17777 0001196B 723F <1> jc short sysenv_err
17778 0001196D 89FE <1> mov esi, edi
17779 0001196F 8A06 <1> mov al, [esi]
17780 00011971 3C20 <1> cmp al, 20h
17781 00011973 7637 <1> jna short sysenv_err
17782 00011975 3C3D <1> cmp al, '='
17783 00011977 7433 <1> je short sysenv_err
17784 00011979 56 <1> push esi
17785 <1> sysenv_1:
17786 0001197A 46 <1> inc esi
17787 0001197B 803E3D <1> cmp byte [esi], '='
17788 0001197E 7433 <1> je short sysenv_3
17789 00011980 803E20 <1> cmp byte [esi], 20h
17790 00011983 73F5 <1> jnb short sysenv_1
17791 00011985 C60600 <1> mov byte [esi], 0
17792 00011988 5E <1> pop esi
17793 <1> ; EDI = Output buffer
17794 <1> ; CX = Buffer length
17795 00011989 30C0 <1> xor al, al
17796 <1> ; AL = 0 -> use ASCIIZ word in [ESI]
17797 <1> ; ESI = Environment variable name address
17798 0001198B E81781FFFF <1> call get_environment_string
17799 00011990 721A <1> jc short sysenv_err
17800 00011992 89DF <1> mov edi, ebx ; * ; user's buffer address
17801 00011994 89C1 <1> mov ecx, eax ; String length
17802 00011996 89E6 <1> mov esi, esp
17803 <1> ; ESI = system buffer address
17804 <1> ; EDI = User's buffer address
17805 <1> ; ECX = transfer (byte) count
17806 00011998 E86EF1FFFF <1> call transfer_to_user_buffer
17807 0001199D 720D <1> jc short sysenv_err
17808 0001199F 890D[548E0100] <1> mov [u.r0], ecx ; transfer (byte) count

```

```

17809 <1> sysenv_2:
17810 000119A5 89EC <1> mov esp, ebp
17811 000119A7 E9D2B0FFFF <1> jmp sysret
17812 <1> sysenv_err:
17813 000119AC 89EC <1> mov esp, ebp
17814 <1> sysenv_error: ; 23/07/2022
17815 000119AE E9ABB0FFFF <1> jmp error
17816 <1> sysenv_3:
17817 000119B3 46 <1> inc esi
17818 000119B4 803E20 <1> cmp byte [esi], 20h
17819 000119B7 73FA <1> jnb short sysenv_3
17820 000119B9 C60600 <1> mov byte [esi], 0
17821 000119BC 5E <1> pop esi
17822 000119BD E8A881FFFF <1> call set_environment_string
17823 000119C2 72E8 <1> jc short sysenv_err
17824 000119C4 8915[548E0100] <1> mov [u.r0], edx
17825 000119CA EBD9 <1> jmp short sysenv_2
17826 <1>
17827 <1> isintr:
17828 <1> ; 21/08/2024 - TRDOS 386 v2.0.9
17829 <1> ; check terminate (CTRL+BREAK)
17830 <1> ; interrupt is enabled or disabled
17831 <1> ; test word [u.intr], 0FFFFh
17832 000119CC F605[A08E0100]FF <1> test byte [u.intr], 0FFh
17833 <1> ; zf = 1 -> terminate interrupt disabled
17834 <1> ; zf = 0 -> terminate interrupt enabled
17835 <1>
17836 <1> ; 19/12/2025
17837 <1> iget:
17838 <1> sndc:
17839 <1> sleep:
17840 000119D3 C3 <1> retn
17841 <1>
17842 <1> ; 23/07/2022 - TRDOS 386 v2.0.5
17843 <1>
17844 <1> ; 22/01/2021
17845 <1> ; temporary - 24/01/2016
17846 <1>
17847 <1> ; 19/12/2025
17848 <1> iget:
17849 <1> ; retn
17850 <1> ; iopen:
17851 <1> ; retn
17852 <1> ; iclose:
17853 <1> ; retn
17854 <1> ; sndc:
17855 <1> ; retn
17856 <1> ; access:
17857 <1> ; retn
17858 <1> ; sleep:
17859 <1> ; retn
3438 <1> %include 'trdosk7.s' ; 24/01/2016
1 <1> *****
2 <1> TRDOS386.ASM (TRDOS 386 Kernel - v2.0.6) - DISK READ&WRITE : trdosk7.s
3 <1> -----
4 <1> Last Update: 29/08/2023 (Previous: 25/07/2022 - Kernel v2.0.5)
5 <1> -----
6 <1> Beginning: 24/01/2016
7 <1> -----
8 <1> Assembler: NASM version 2.11 (trdos386.s)
9 <1> -----
10 <1> Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
11 <1> DISK_IO.ASM (20/07/2011)
12 <1> *****
13 <1> DISK_IO.ASM (c) 2009-2011 Erdogan TAN [ 04/07/2009 ] Last Update: 20/07/2011
14 <1>
15 <1> disk_write:
16 <1> ; 25/02/2016
17 <1> ; 24/02/2016
18 <1> ; 23/02/2016
19 000119D4 807E0500 <1> cmp byte [esi+LD_LBAYes], 0
20 000119D8 7774 <1> ja short lba_write
21 <1>
22 <1> chs_write:
23 <1> ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
24 <1> ; 25/02/2016
25 <1> ; 23/02/2016
26 000119DA C605[517F0100]03 <1> mov byte [disk_rw_op], 3 ; CHS write
27 000119E1 EB0D <1> jmp short chs_rw
28 <1>
29 <1> disk_read:
30 <1> ; 25/02/2016
31 <1> ; 24/02/2016
32 <1> ; 23/02/2016
33 <1> ; 17/02/2016
34 <1> ; 14/02/2016
35 <1> ; 31/01/2016 (TRDOS 386 = TRDOS v2.0)
36 <1> ; 17/10/2010
37 <1> ; 18/04/2010
38 <1> ;
39 <1> ; INPUT -> EAX = Logical Block Address
40 <1> ; ESI = Logical Dos Disk Table offset (DRV)
41 <1> ; ECX = Sector Count
42 <1> ; EBX = Destination Buffer
43 <1> ; OUTPUT ->
44 <1> ; cf = 0 or cf = 1
45 <1> ; (Modified registers: EAX, EBX, ECX, EDX)
46 <1>
47 000119E3 807E0500 <1> cmp byte [esi+LD_LBAYes], 0
48 000119E7 776E <1> ja short lba_read
49 <1>
50 <1> chs_read:
51 <1> ; 29/08/2023 (TRDOS 386 Kernel v2.0.6)
52 <1> ; 25/07/2022 (TRDOS 386 Kernel v2.0.5)
53 <1> ; 25/02/2016
54 <1> ; 24/02/2016
55 <1> ; 23/02/2016
56 <1> ; 31/01/2016 (TRDOS 386 = TRDOS v2.0)
57 <1> ; 20/07/2011
58 <1> ; 04/07/2009
59 <1> ;
60 <1> ; INPUT -> EAX = Logical Block Address
61 <1> ; ECX = Number of sectors to read
62 <1> ; ESI = Logical Dos Disk Table Offset (DRV)
63 <1> ; EBX = Destination Buffer
64 <1> ; OUTPUT ->
65 <1> ; cf = 0 or cf = 1
66 <1> ; (Modified registers: EAX, EBX, ECX, EDX)
67 <1>
68 <1> ; 23/02/2016
69 000119E9 C605[517F0100]02 <1> mov byte [disk_rw_op], 2 ; CHS read
70 <1>
71 <1> chs_rw:
72 <1> ; movzx edx, word [esi+LD_BPB+SecPerTrack]

```



```

73      <1>      ;movzx edx, byte [esi+LD_BPB+SecPerTrack] ; <= 63
74      <1>      ;mov      [disk_rw_spt], dl
75      <1>
76      <1>      chs_read_next_sector:
77 000119F0 C605[527F0100]04 <1>      mov      byte [retry_count], 4
78      <1>
79      <1>      chs_read_retry:
80      <1>      ;mov      [sector_count], ecx ; 23/02/2016
81      <1>
82 000119F7 50 <1>      push     eax                ; Linear sector #
83 000119F8 51 <1>      push     ecx                ; # of FAT/FILE/DIR sectors
84      <1>
85 000119F9 0FB74E1E <1>      movzx    ecx, word [esi+LD_BPB+SecPerTrack]
86      <1>      ;movzx    ecx, byte [disk_rw_spt] ; 23/02/2016
87 000119FD 29D2 <1>      sub      edx, edx
88 000119FF F7F1 <1>      div      ecx
89      <1>      ; eax = track, dx (dl) = sector (on track)
90      <1>      ;sub      cl, dl ; 24/02/2016 (spt - sec)
91      <1>      ;push     ecx ; *
92      <1>      ; 25/07/2022
93      <1>      ;mov      cx, dx                ; Sector (zero based)
94      <1>      ;inc      cx                ; To make it 1 based
95      <1>      ;push     cx
96      <1>      ; 29/08/2023
97      <1>      ;mov      ecx, edx
98      <1>      ;inc      ecx
99      <1>      ;push     ecx
100 00011A01 42 <1>      inc      edx
101 00011A02 52 <1>      push     edx ; dl = sector number (on track)
102      <1>      ;
103 00011A03 668B4E20 <1>      mov      cx, [esi+LD_BPB+Heads]
104      <1>      ;sub      dx, dx
105      <1>      ; 25/07/2022
106 00011A07 29D2 <1>      sub      edx, edx
107 00011A09 F7F1 <1>      div      ecx
108      <1>      ; eax (ax) = cylinder, dx (dl) = head (max. FFh)
109 00011A0B 88D6 <1>      mov      dh, dl
110      <1>      ;pop      cx                ; AX=Cyl, DH=Head, CX=Sector
111      <1>      ; 25/07/2022
112 00011A0D 59 <1>      pop      ecx ; sector number (on track)
113 00011A0E 8A5602 <1>      mov      dl, [esi+LD_PhyDrvNo]
114      <1>
115 00011A11 88C5 <1>      mov      ch, al                ; NOTE: max. 1023 cylinders !
116 00011A13 C0CC02 <1>      ror      ah, 2                ; Rotate 2 bits right
117 00011A16 08E1 <1>      or       cl, ah
118      <1>
119      <1>      ; 24/02/2016
120      <1>      ;pop      eax ; * (spt - sec) (example: 63 - 0 = 63)
121      <1>      ;cmp      eax, [sector_count]
122      <1>      ;jb      short chs_write_sectors
123      <1>      ;je      short chs_read_sectors
124      <1>      ; ; (# of sectors to read is more than remaining sectors on the track)
125      <1>      ;mov      al, [sector_count]
126      <1>      ;chs_read_sectors: ; read or write !
127 00011A18 B001 <1>      mov      al, 1 ; 25/02/2016
128 00011A1A 8A25[517F0100] <1>      mov      ah, [disk_rw_op] ; 02h = chs read, 03h = chs write
129      <1>      ;
130 00011A20 E8E635FFFF <1>      call     int13h                ; BIOS Service func ( ah ) = 2
131      <1>      ;
132      <1>      ; Read disk sectors
133      <1>      ; AL-sec num CH-track CL-sec
134      <1>      ; DH-head DL-drive ES:BX-buffer
135      <1>      ; CF-flag AH-stat AL-sec read
136      <1>      ; If CF = 1 then (If AH > 0)
137 00011A25 8825[537F0100] <1>      mov      [disk_rw_err], ah
138      <1>
139 00011A2B 59 <1>      pop      ecx
140 00011A2C 58 <1>      pop      eax
141 00011A2D 7314 <1>      jnc      short chs_read_ok
142      <1>
143 00011A2F 803D[537F0100]09 <1>      cmp      byte [disk_rw_err], 09h ; DMA crossed 64K segment boundary
144      <1>      je      short chs_read_error_retn
145 00011A38 FE0D[527F0100] <1>      dec      byte [retry_count]
146 00011A3E 75B7 <1>      jnz      short chs_read_retry
147      <1>
148      <1>      chs_read_error_retn:
149 00011A40 F9 <1>      stc
150      <1>      ;retn
151 00011A41 EB67 <1>      jmp      short update_drv_error_byte
152      <1>
153      <1>      ;chs_write_sectors: ; read or write
154      <1>      ; ; (# of sectors to read is less than remaining sectors on the track)
155      <1>      ;mov      [sector_count], al
156      <1>      ;jmp      short chs_read_sectors
157      <1>
158      <1>      chs_read_ok:
159      <1>      ; ; 23/02/2016
160      <1>      ;movzx    edx, byte [sector_count] ; sector count (<= spt)
161      <1>      ;sub      ecx, edx ; remaining sector count
162      <1>      ;jna      short update_drv_error_byte
163      <1>      ;add      eax, edx ; next disk sector
164      <1>      ;shl      edx, 9 ; 512 * sector count
165      <1>      ;add      ebx, edx ; next buffer byte address
166      <1>      ;jmp      chs_read_next_sector
167      <1>      ; 25/02/2016
168 00011A43 40 <1>      inc      eax ; next sector
169 00011A44 81C300020000 <1>      add      ebx, 512
170 00011A4A E2A4 <1>      loop     chs_read_next_sector
171 00011A4C EB5C <1>      jmp      short update_drv_error_byte
172      <1>
173      <1>      lba_write:
174      <1>      ; 23/02/2016
175 00011A4E C605[517F0100]1C <1>      mov      byte [disk_rw_op], 1Ch ; LBA write
176 00011A55 EB07 <1>      jmp      short lba_rw
177      <1>
178      <1>      lba_read:
179      <1>      ; 14/07/2022
180      <1>      ; 23/02/2016
181      <1>      ; 17/02/2016
182      <1>      ; 14/02/2016
183      <1>      ; 13/02/2016
184      <1>      ; 31/01/2016 (TRDOS 386 = TRDOS v2.0)
185      <1>      ; 10/07/2015 (Retro UNIX 386 v1)
186      <1>      ;
187      <1>      ; INPUT -> EAX = Logical Block Address
188      <1>      ; ESI = Logical Dos Disk Table offset (DRV)
189      <1>      ; ECX = Sector Count
190      <1>      ; EBX = Destination Buffer
191      <1>      ; OUTPUT ->
192      <1>      ; cf = 0 or cf = 1
193      <1>      ; (Modified registers: EAX, EBX, ECX, EDX)
194      <1>
195      <1>      ; LBA read/write (with private LBA function)
196      <1>      ; ((Retro UNIX 386 v1 - DISK I/O code by Erdogan Tan))

```

```

197 <1>
198 <1> ; 23/02/2016
199 00011A57 C605[517F0100]1B <1> mov byte [disk_rw_op], 1Bh ; LBA read
200 <1>
201 <1> lba_rw:
202 <1> ; 17/02/2016
203 00011A5E 57 <1> push edi
204 <1>
205 00011A5F 890D[547F0100] <1> mov [sector_count], ecx ; total sector (read) count
206 <1>
207 00011A65 8A5602 <1> mov dl, [esi+LD_PhyDrvNo]
208 <1> ; dl = physical drive number (0, 1, 80h, 81h, 82h, 83h)
209 <1>
210 <1> lba_read_next:
211 <1> ; 14/07/2022
212 00011A68 BF00010000 <1> mov edi, 256
213 00011A6D 39F9 <1> cmp ecx, edi
214 <1> ; cmp ecx, 256
215 00011A6F 7602 <1> jna short lba_read_rsc
216 <1> ; mov ecx, 256 ; 17/02/2016
217 00011A71 89F9 <1> mov ecx, edi
218 <1> lba_read_rsc:
219 00011A73 290D[547F0100] <1> sub [sector_count], ecx ; remain sectors
220 <1>
221 00011A79 89CF <1> mov edi, ecx
222 00011A7B 89C1 <1> mov ecx, eax ; sector number/address
223 <1>
224 00011A7D C605[527F0100]04 <1> mov byte [retry_count], 4
225 <1> lba_read_retry:
226 00011A84 89F8 <1> mov eax, edi
227 <1> ;
228 <1> ; ecx = sector number
229 <1> ; al = sector count (0 - 255) /// (0 = 256)
230 <1> ; dl = drive number
231 <1> ; ebx = buffer offset
232 <1> ;
233 <1> ; Function 1Bh = LBA read, 1Ch = LBA write
234 <1> ; 23/02/2016
235 00011A86 8A25[517F0100] <1> mov ah, [disk_rw_op] ; 1Bh = LBA read, 1Ch = LBA write
236 00011A8C E87A35FFFF <1> call int13h
237 <1> ; al = ? (changed)
238 <1> ; ah = error code
239 00011A91 8825[537F0100] <1> mov [disk_rw_err], ah
240 00011A97 7332 <1> jnc short lba_read_ok
241 00011A99 80FC80 <1> cmp ah, 80h ; time out?
242 00011A9C 740A <1> je short lba_read_stc_retn
243 00011A9E FE0D[527F0100] <1> dec byte [retry_count]
244 00011AA4 7FDE <1> jg short lba_read_retry
245 00011AA6 7438 <1> jz short lba_read_reset
246 <1> ; sf = 1
247 <1>
248 <1> lba_read_stc_retn:
249 00011AA8 F9 <1> stc
250 <1> lba_read_retn:
251 00011AA9 5F <1> pop edi
252 <1>
253 <1> update_drv_error_byte:
254 00011AAA 9C <1> pushf
255 00011AAB 53 <1> push ebx
256 <1> ; push cx
257 00011AAC 51 <1> push ecx ; 14/07/2022
258 <1> ; or ecx, ecx
259 <1> ; jz short udrv_errb0
260 00011AAD 8A0D[537F0100] <1> mov cl, [disk_rw_err]
261 <1> udrv_errb0:
262 00011AB3 0FB65E02 <1> movzx ebx, byte [esi+LD_PhyDrvNo]
263 00011AB7 80FB02 <1> cmp bl, 2
264 00011ABA 7203 <1> jb short udrv_errb1
265 00011ABC 80EB7E <1> sub bl, 7Eh
266 <1> ; cmp bl, 5
267 <1> ; ja short udrv_errb2
268 <1> udrv_errb1:
269 00011ABF 81C3[A5660000] <1> add ebx, drv.error ; 13/02/2016
270 00011AC5 880B <1> mov [ebx], cl ; error code
271 <1> udrv_errb2:
272 <1> ; pop cx
273 00011AC7 59 <1> pop ecx ; 14/07/2022
274 00011AC8 5B <1> pop ebx
275 00011AC9 9D <1> popf
276 00011ACA C3 <1> retn
277 <1>
278 <1> lba_read_ok:
279 00011ACB 89C8 <1> mov eax, ecx ; sector number
280 00011ACD 01F8 <1> add eax, edi ; sector number (next)
281 00011ACF C1E709 <1> shl edi, 9 ; sector count * 512
282 00011AD2 01FB <1> add ebx, edi ; next buffer offset
283 <1>
284 00011AD4 8B0D[547F0100] <1> mov ecx, [sector_count] ; remaining sectors
285 00011ADA 09C9 <1> or ecx, ecx
286 00011ADC 758A <1> jnz short lba_read_next
287 00011ADE EBC9 <1> jmp short lba_read_retn
288 <1>
289 <1> lba_read_reset:
290 00011AE0 B40D <1> mov ah, 0Dh ; Alternate reset
291 00011AE2 E82435FFFF <1> call int13h
292 <1> ; al = ? (changed)
293 <1> ; ah = error code
294 00011AE7 739B <1> jnc short lba_read_retry
295 00011AE9 EBBE <1> jmp short lba_read_retn
3439 <1> %include 'trdosk8.s' ; 24/01/2016
1 <1> ; *****
2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.10) - MAIN PROGRAM : trdosk8.s
3 <1> ;
4 <1> ; Last Update: 28/01/2025 (Previous: 29/12/2024)
5 <1> ;
6 <1> ; Beginning: 24/01/2016
7 <1> ;
8 <1> ; Assembler: NASM version 2.15 (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> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
20 <1> ; 23/12/2016
21 <1> ; 10/06/2016
22 <1> ; 22/05/2016
23 <1> ; 20/05/2016
24 <1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)

```

```

25      <1> ; TRDOS 386 feature only !
26      <1>
27      <1> INPUT ->
28      <1> AL = process number (next process)
29      <1>
30      <1> this process must be added to run sequence
31      <1>
32      <1> [u.pri] = priority of present process
33      <1>
34      <1> DL = priority (queue)
35      <1> 0 = background (low) ; run on background
36      <1> 1 = regular (normal) ; run as regular
37      <1> 2 = event (high) ; run for event
38      <1>
39      <1> 1) If the requested process is already running:
40      <1> a) If present priority is high ([u.pri]=2)
41      <1> and requested priority is also high,
42      <1> there is nothing to do! Because it has been
43      <1> done already (before this attempt).
44      <1> b) If present priority is high ([u.pri]=2)
45      <1> and requested priority is not high, there is
46      <1> nothing to do! Because, it's current
47      <1> run queue is unspecified, here. (It may be in
48      <1> a waiting list or in a run queue; if the new
49      <1> priority would be used to add it to relevant
50      <1> run queue, this would be wrong, unnecessary
51      <1> and destabilizing duplication!)
52      <1> c) If present priority is not high ([u.pri]<2)
53      <1> and requested priority is high (event),
54      <1> process will be added to present priority's
55      <1> run queue and then, priority will be changed
56      <1> to high ([u.pri]=2).
57      <1> d) If present priority is not high ([u.pri]<2)
58      <1> and requested priority is not high, [u.pri]
59      <1> value will be changed. There is nothing to do
60      <1> in addition. (The new priority value will be
61      <1> used by 'tswap/tswitch' procedure at 'sysret'
62      <1> or 'sysrele' stage.)
63      <1>
64      <1> 2) If the requested process is not running:
65      <1> a) If requested priority of the requested
66      <1> (next) process is high (event) and priority
67      <1> of present process is not high, the requested
68      <1> process will be added to ('runq_event') high
69      <1> priority run queue and then present (running)
70      <1> process will be stopped (swapped/switched out)
71      <1> immediately if it is in user mode, or it's
72      <1> [u.quant] value will be reset to 0 and (then)
73      <1> it will be stopped at 'sysret' stage.
74      <1> b) If requested priority of the requested
75      <1> (next) process is high (event) and priority
76      <1> of present process is also high, the requested
77      <1> process will be added to ('runq_event') high
78      <1> priority run queue and present (running)
79      <1> process will be allowed to run until it's
80      <1> time quantum will be elapsed ([u.quant]=0).
81      <1> c) If requested priority of the requested
82      <1> (next) process is not high ('run for event'),
83      <1> there is nothing to do. Because, it's current
84      <1> run queue is unspecified, here. (It may be in
85      <1> a waiting list or in a run queue; if the new
86      <1> priority would be used to add it to relevant
87      <1> run queue, this would be wrong, unnecessary
88      <1> and destabilizing duplication!)
89      <1>
90      <1> OUTPUT ->
91      <1> none
92      <1>
93      <1> [u.pri] = priority of present process
94      <1>
95      <1> cf = 1, if the request could not be fulfilled.
96      <1>
97      <1> NOTE:
98      <1> ; * Processes in 'run as regular' queue can run
99      <1> ; if there is no process in 'run for event' queue
100      <1> ; ('run for event' processes have higher priority)
101      <1> ; * When [u.quant] time quantum of a process is
102      <1> ; elapsed, it's high priority ('run for event')
103      <1> ; status will be disabled, it can be run in sequence
104      <1> ; of it's actual run queue.
105      <1> ; * A 'run on background' process will always be
106      <1> ; sequenced in 'run on background' (low priority)
107      <1> ; queue, it can run only when other priority queues
108      <1> ; are empty. (idle time processes, e.g. printing)
109      <1>
110      <1> Modified registers: eax, ebx, edx
111      <1>
112      <1>
113      <1> srunseq_0:
114      <1> cmp al, [u.uno] ; same process ?
115      <1> jne short srunseq_2 ; no
116      <1>
117      <1> mov ah, [u.pri] ; present/current priority
118      <1> cmp ah, 2 ; 'run for event' priority level
119      <1> jb short srunseq_6 ; no
120      <1>
121      <1> srunseq_1:
122      <1> ; there is nothing to do!
123      <1> ret
124      <1>
125      <1> srunseq_2:
126      <1> ; this not necessary ! 23/12/2016
127      <1> ; cmp al, nproc ; number of processes = 16
128      <1> ; jnb short srunseq_5 ; error ! invalid process number
129      <1>
130      <1> ; dl = priority
131      <1> cmp dl, 2 ; event queue
132      <1> jb short srunseq_1 ; requested process is not present
133      <1> ; process and priority of requested
134      <1> ; process is not high (event),
135      <1> ; there is nothing to do!
136      <1>
137      <1> ; requested process is not present process
138      <1> ; & priority of requested process is high
139      <1> cmp dl, [u.pri] ; priority of present process
140      <1> jna short srunseq_3 ; is high, also
141      <1>
142      <1> ; present process will be swapped/switched out
143      <1> inc byte [p_change] ; 1
144      <1>
145      <1> srunseq_3:
146      <1> ; add process to 'runq_event' queue for new event
147      <1> mov ebx, runq_event ; high priority run queue
148      <1>

```

```

149      <1> srunseq_4:
150      <1> ; al = process number
151      <1> ; ebx = run queue
152      <1> ; call putlu
153      <1> ; retn
154      <1> ; 29/07/2022
155 00011B17 E91BEFFFFF <1> jmp putlu
156      <1>
157      <1> srunseq_5:
158 00011B1C F5 <1> cmc
159 00011B1D C3 <1> retn
160      <1>
161      <1> srunseq_6:
162      <1> ; present priority of the process is not high
163      <1>
164 00011B1E 8815[9E8E0100] <1> mov [u.pri], dl ; new priority
165      <1> ; (will be used by 'tswap')
166      <1>
167 00011B24 80FA02 <1> cmp dl, 2 ; high priority ?
168 00011B27 72F3 <1> jb short srunseq_5 ; no, there is nothing to do
169      <1> ; in addition
170      <1>
171      <1> ; process must be added to relevant run queue, here!
172      <1> ; (new priority is high/event priority and process
173      <1> ; will not be added to a run queue by 'tswap')
174      <1>
175 00011B29 BB[448E0100] <1> mov ebx, runq_normal ; 'run as regular' queue
176      <1>
177 00011B2E 20E4 <1> and ah, ah ; previous value of [u.pri]
178 00011B30 75E5 <1> jnz short srunseq_4
179      <1>
180 00011B32 43 <1> inc ebx
181 00011B33 43 <1> inc ebx
182      <1> ; ebx = runq_background ; 'run on backgroud' queue
183      <1>
184 00011B34 EBE1 <1> jmp short srunseq_4
185      <1> clock:
186      <1> ; 21/08/2024 - TRDOS 386 v2.0.9
187      <1> ; 23/05/2016
188      <1> ; 22/05/2016
189      <1> ; 20/05/2016
190      <1> ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
191      <1> ; 14/05/2015 - 14/10/2015 (Retro UNIX 386 v1)
192      <1> ; 07/12/2013 - 10/04/2014 (Retro UNIX 8086 v1)
193      <1>
194      <1> ;;;
195      <1> ; 21/08/2024
196      <1> ; cmp word [u.quit], 0FFFFh
197 00011B36 803D[A28E0100]FF <1> cmp byte [u.quit], 0FFh
198 00011B3D 721A <1> jb short clk_0
199      <1> ; CTRL+BRK keys pressed
200      <1> ; cmp word [u.intr], 0
201 00011B3F 803D[A08E0100]00 <1> cmp byte [u.intr], 0
202 00011B46 7611 <1> jna short clk_0
203 00011B48 C605[31830100]FF <1> mov byte [p_change], 0FFh ; -1 ; CTRL+BREAK sign
204 00011B4F F605[9C8E0100]FF <1> test byte [u.quant], 0FFh ; if [u.quant] number > 0
205 00011B56 752D <1> jnz short clk_1 ; decrease [u.quant] number
206      <1> ; [u.quant] = 0
207 00011B58 C3 <1> retn
208      <1> ;;;
209      <1> clk_0:
210 00011B59 803D[9C8E0100]00 <1> cmp byte [u.quant], 0
211 00011B60 7723 <1> ja short clk_1
212      <1> ;
213 00011B62 803D[A58E0100]01 <1> cmp byte [u.uno], 1 ; /etc/init ? (for Retro UNIX 8086 & 386 v1)
214      <1> ; MainProg (Kernel's Command Interpreter)
215      <1> ; for TRDOS 386.
216 00011B69 761A <1> jna short clk_1 ; yes, do not swap out
217      <1> ;
218 00011B6B 803D[488E0100]FF <1> cmp byte [sysflg], 0FFh ; user or system space ?
219 00011B72 7517 <1> jne short clk_2 ; system space (sysflg <> 0FFh)
220      <1> ;
221      <1> ; 21/08/2024
222      <1> ; cmp word [u.intr], 0
223      <1> ; jna short clk_2
224      <1> ;
225      <1> ; 23/05/2016
226 00011B74 803D[32830100]00 <1> cmp byte [multi_tasking], 0
227 00011B7B 760E <1> jna short clk_2
228      <1> ;
229      <1> ; 21/08/2024
230      <1> ; inc byte [p_change] ; it is time to change running process
231 00011B7D C605[31830100]01 <1> mov byte [p_change], 1
232 00011B84 C3 <1> retn
233      <1> clk_1:
234 00011B85 FE0D[9C8E0100] <1> dec byte [u.quant]
235      <1> clk_2:
236 00011B8B C3 <1> retn ; return to (hardware) timer interrupt routine
237      <1>
238      <1> ; 12/10/2017
239      <1> ; 15/01/2017
240      <1> ; 14/01/2017
241      <1> ; 07/01/2017
242      <1> ; 02/01/2017
243      <1> ; 17/08/2016
244      <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
245      <1> int34h: ; #IOCTL# (I/O port access support for ring 3)
246      <1> ;
247      <1> ; 23/05/2016
248      <1> ; 20/06/2016
249      <1> ; 29/04/2016 - TRDOS 386 (TRDOS v2.0)
250      <1> ;
251      <1> ; INPUT ->
252      <1> ; AH = 0 -> read port (physical IO port) -byte-
253      <1> ; AH = 1 -> write port (physical IO port) -byte-
254      <1> ; AL = data byte
255      <1> ; AH = 2 -> read port (physical IO port) -word-
256      <1> ; AH = 3 -> write port (physical IO port) -word-
257      <1> ; BX = data word
258      <1> ; AH = 4 -> read port (physical IO port) -dword-
259      <1> ; AH = 5 -> write port (physical IO port) -dword-
260      <1> ; EBX = data dword
261      <1> ; 12/10/2017
262      <1> ; AH = 6 -> read port (physical IO port) twice -byte-
263      <1> ; AH = 7 -> write port (physical IO port) twice -byte-
264      <1> ; BX = data word
265      <1> ;
266      <1> ; DX = Port number (<= 0FFFFh)
267      <1> ;
268      <1> ; OUTPUT ->
269      <1> ; AL = data byte (in al, dx)
270      <1> ; AX = data word (in ax, dx)
271      <1> ; EAX = data dword (in eax, dx)
272      <1> ;

```

```

273      <1>      ;      (ECX = actual TRANSFER COUNT for string functions)
274      <1>      ;
275      <1>      ;
276      <1>      ; Modified registers: EAX
277      <1>      ;
278      <1>      ;
279      <1>      ;cmp    ah, 5
280      <1>      ;ja     short int34h_5 ; invalid function !
281      <1>      ;
282      <1>      ; 12/10/2017
283      <1>      cmp     ah, 7
284      <1>      ja     short int34h_5 ; invalid function !
285      <1>      ;
286      <1>      ;; 15/01/2017
287      <1>      ;; 14/01/2017
288      <1>      ;; 02/01/2017
289      <1>      ;;mov   byte [ss:intf1g], 34h ; IOCTL interrupt
290      <1>      sti
291      <1>      ;
292      <1>      ;sti    ; enable interrupts
293      <1>      and     byte [esp+8], 11111110b      ; clear carry bit of eflags register
294      <1>      ;
295      <1>      cmp     ah, 1
296      <1>      jnb    short int34h_0
297      <1>      ja     short int34h_1
298      <1>      ;
299      <1>      out     dx, al
300      <1>      ;iretd
301      <1>      jmp     short int34h_iret
302      <1>      ;
303      <1>      int34h_0:
304      <1>      in      al, dx
305      <1>      ;iretd
306      <1>      int34h_iret:
307      <1>      ;cli    ; 07/01/2017
308      <1>      ;; 15/01/2017
309      <1>      ;;mov   byte [ss:intf1g], 0 ; reset
310      <1>      ;iretd
311      <1>      ;
312      <1>      int34h_1:
313      <1>      test    ah, 1
314      <1>      jnz     short int34h_3 ; out
315      <1>      ;
316      <1>      ; in
317      <1>      cmp     ah, 2
318      <1>      ja     short int34h_2
319      <1>      ;
320      <1>      mov     ax, bx
321      <1>      in      ax, dx
322      <1>      ;iretd
323      <1>      jmp     short int34h_iret
324      <1>      ;
325      <1>      int34h_2:
326      <1>      cmp     ah, 4
327      <1>      ja     short int34h_7 ; 12/10/2017
328      <1>      ; ah = 4
329      <1>      mov     eax, ebx
330      <1>      in      eax, dx
331      <1>      ;iretd
332      <1>      jmp     short int34h_iret
333      <1>      ;
334      <1>      int34h_3:
335      <1>      cmp     ah, 3
336      <1>      ja     short int34h_4
337      <1>      ;
338      <1>      mov     ax, bx
339      <1>      out     dx, ax
340      <1>      ;iretd
341      <1>      jmp     short int34h_iret
342      <1>      ;
343      <1>      int34h_4:
344      <1>      cmp     ah, 5
345      <1>      ja     short int34h_6 ; 12/10/2017
346      <1>      ; ah = 5
347      <1>      mov     eax, ebx
348      <1>      out     dx, eax
349      <1>      ;iretd
350      <1>      jmp     short int34h_iret
351      <1>      ;
352      <1>      int34h_5:
353      <1>      or      byte [esp+8], 1      ; set carry bit of eflags register
354      <1>      ;iretd
355      <1>      ;
356      <1>      ; 12/10/2017
357      <1>      int34h_6:
358      <1>      mov     ax, bx
359      <1>      out     dx, al
360      <1>      jmp     short $+2
361      <1>      xchg    ah, al
362      <1>      out     dx, al
363      <1>      ;xchg    al, ah
364      <1>      ;iretd
365      <1>      jmp     short int34h_8
366      <1>      int34h_7:
367      <1>      in      al, dx
368      <1>      jmp     short $+2
369      <1>      mov     ah, al
370      <1>      in      al, dx
371      <1>      int34h_8:
372      <1>      xchg    al, ah
373      <1>      ;iretd
374      <1>      ;
375      <1>      INT4Ah:
376      <1>      ; 24/01/2016
377      <1>      ; this procedure will be called by 'RTC_INT' (in 'timer.s')
378      <1>      retn
379      <1>      ;
380      <1>      ; u0.s
381      <1>      ; Retro UNIX 386 v1 kernel (v0.2) - SYS0.INC
382      <1>      ; Last Modification: 20/11/2015
383      <1>      ;
384      <1>      com2_int:
385      <1>      ; 07/11/2015
386      <1>      ; 24/10/2015
387      <1>      ; 23/10/2015
388      <1>      ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)
389      <1>      ; 28/07/2014 (Retro UNIX 8086 v1)
390      <1>      ; < serial port 2 interrupt handler >
391      <1>      ;
392      <1>      mov     [esp], eax ; overwrite call return address
393      <1>      ;push    eax
394      <1>      mov     ax, 9
395      <1>      jmp     short comm_int
396      <1>      com1_int:

```

```

397      <1>      ; 07/11/2015
398      <1>      ; 24/10/2015
399 00011BF8 890424      <1>      mov     [esp], eax ; overwrite call return address
400      <1>      ; 23/10/2015
401      <1>      ;push    eax
402 00011BFB 66B80800    <1>      mov     ax, 8
403      <1>      comm_int:
404      <1>      ; 20/11/2015
405      <1>      ; 18/11/2015
406      <1>      ; 17/11/2015
407      <1>      ; 16/11/2015
408      <1>      ; 09/11/2015
409      <1>      ; 08/11/2015
410      <1>      ; 07/11/2015
411      <1>      ; 06/11/2015 (serial4.asm, 'serial')
412      <1>      ; 01/11/2015
413      <1>      ; 26/10/2015
414      <1>      ; 23/10/2015
415 00011BFF 53          <1>      push    ebx
416 00011C00 56          <1>      push    esi
417 00011C01 57          <1>      push    edi
418 00011C02 1E          <1>      push    ds
419 00011C03 06          <1>      push    es
420      <1>      ; 18/11/2015
421 00011C04 0F20DB      <1>      mov     ebx, cr3
422 00011C07 53          <1>      push    ebx ; ****
423      <1>      ;
424 00011C08 51          <1>      push    ecx ; ***
425 00011C09 52          <1>      push    edx ; **
426      <1>      ;
427 00011C0A BB10000000    <1>      mov     ebx, KDATA
428 00011C0F 8EDB        <1>      mov     ds, bx
429 00011C11 8EC3        <1>      mov     es, bx
430      <1>      ;
431 00011C13 8B0D[A0760100] <1>      mov     ecx, [k_page_dir]
432 00011C19 0F22D9      <1>      mov     cr3, ecx
433      <1>      ; 20/11/2015
434      <1>      ; Interrupt identification register
435 00011C1C 66BAFA02    <1>      mov     dx, 2FAh ; COM2
436      <1>      ;
437 00011C20 3C08        <1>      cmp     al, 8
438 00011C22 7702        <1>      ja     short com_i0
439      <1>      ;
440      <1>      ; 20/11/2015
441      <1>      ; 17/11/2015
442      <1>      ; 16/11/2015
443      <1>      ; 15/11/2015
444      <1>      ; 24/10/2015
445      <1>      ; 14/03/2015 (Retro UNIX 386 v1 - Beginning)
446      <1>      ; 28/07/2014 (Retro UNIX 8086 v1)
447      <1>      ; < serial port 1 interrupt handler >
448      <1>      ;
449 00011C24 FEC6        <1>      inc     dh ; 3FAh ; COM1 Interrupt id. register
450      <1>      com_i0:
451      <1>      ;push    eax ; *
452      <1>      ; 07/11/2015
453 00011C26 A2[0A770100] <1>      mov     byte [ccomport], al
454      <1>      ; 09/11/2015
455 00011C2B 0FB7D8      <1>      movzx   ebx, ax ; 8 or 9
456      <1>      ; 17/11/2015
457      <1>      ; reset request for response status
458 00011C2E 88A3[00770100] <1>      mov     [ebx+req_resp-8], ah ; 0
459      <1>      ;
460      <1>      ; 20/11/2015
461 00011C34 EC          <1>      in      al, dx          ; read interrupt id. register
462 00011C35 EB00        <1>      JMP     $+2          ; I/O DELAY
463 00011C37 2404        <1>      and     al, 4          ; received data available?
464 00011C39 7470        <1>      jz     short com_eoi ; (transmit. holding reg. empty)
465      <1>      ;
466      <1>      ; 20/11/2015
467 00011C3B 80EA02      <1>      sub     dl, 3FAh-3F8h ; data register (3F8h, 2F8h)
468 00011C3E EC          <1>      in      al, dx          ; read character
469      <1>      ;JMP     $+2          ; I/O DELAY
470      <1>      ; 08/11/2015
471      <1>      ; 07/11/2015
472 00011C3F 89DE        <1>      mov     esi, ebx
473 00011C41 89DF        <1>      mov     edi, ebx
474 00011C43 81C6[04770100] <1>      add     esi, rchar - 8 ; points to last received char
475 00011C49 81C7[06770100] <1>      add     edi, schar - 8 ; points to last sent char
476 00011C4F 8806        <1>      mov     [esi], al ; received char (current char)
477      <1>      ; query
478 00011C51 20C0        <1>      and     al, al
479 00011C53 7527        <1>      jnz     short com_i2
480      <1>      ; response
481      <1>      ; 17/11/2015
482      <1>      ; set request for response status
483 00011C55 FE83[00770100] <1>      inc     byte [ebx+req_resp-8] ; 1
484      <1>      ;
485 00011C5B 6683C205    <1>      add     dx, 3FDh-3F8h ; (3FDh, 2FDh)
486 00011C5F EC          <1>      in      al, dx          ; read line status register
487 00011C60 EB00        <1>      JMP     $+2          ; I/O DELAY
488 00011C62 2420        <1>      and     al, 20h        ; transmitter holding reg. empty?
489 00011C64 7445        <1>      jz     short com_eoi ; no
490 00011C66 B0FF        <1>      mov     al, 0FFh      ; response
491 00011C68 6683EA05    <1>      sub     dx, 3FDh-3F8h ; data port (3F8h, 2F8h)
492 00011C6C EE          <1>      out     dx, al         ; send on serial port
493      <1>      ; 17/11/2015
494 00011C6D 803F00      <1>      cmp     byte [edi], 0 ; query ? (schar)
495 00011C70 7502        <1>      jne     short com_i1 ; no
496 00011C72 8807        <1>      mov     [edi], al      ; 0FFh (responded)
497      <1>      com_i1:
498      <1>      ; 17/11/2015
499      <1>      ; reset request for response status (again)
500 00011C74 FE8B[00770100] <1>      dec     byte [ebx+req_resp-8] ; 0
501 00011C7A EB2F        <1>      jmp     short com_eoi
502      <1>      com_i2:
503      <1>      ; 08/11/2015
504 00011C7C 3CFF        <1>      cmp     al, 0FFh      ; (response ?)
505 00011C7E 7417        <1>      je     short com_i3 ; (check for response signal)
506      <1>      ; 07/11/2015
507 00011C80 3C04        <1>      cmp     al, 04h ; EOT
508 00011C82 751C        <1>      jne     short com_i4
509      <1>      ; EOT = 04h (End of Transmit) - 'CTRL + D'
510      <1>      ; (an EOT char is supposed as a ctrl+brk from the terminal)
511      <1>      ; 08/11/2015
512      <1>      ; pty -> tty 0 to 7 (pseudo screens)
513 00011C84 861D[CE760100] <1>      xchg    bl, [ptty] ; tty number (8 or 9)
514 00011C8A E89F51FFFF    <1>      call    ctrlbrk
515 00011C8F 861D[CE760100] <1>      xchg    [ptty], bl ; (restore pty value and BL value)
516      <1>      ;mov     al, 04h ; EOT
517      <1>      ; 08/11/2015
518 00011C95 EB09        <1>      jmp     short com_i4
519      <1>      com_i3:
520      <1>      ; 08/11/2015

```

```

521      <1>      ; If 0FFh has been received just after a query
522      <1>      ; (schar, ZERO), it is a response signal.
523      <1>      ; 17/11/2015
524 00011C97 803F00      <1>      cmp     byte [edi], 0 ; query ? (schar)
525 00011C9A 7704      <1>      ja      short com_i4 ; no
526      <1>      ; reset query status (schar)
527 00011C9C 8807      <1>      mov     [edi], al ; 0FFh
528 00011C9E FEC0      <1>      inc     al ; 0
529      <1>      com_i4:
530      <1>      ; 27/07/2014
531      <1>      ; 09/07/2014
532 00011CA0 D0E3      <1>      shl     bl, 1
533 00011CA2 81C3[D0760100] <1>      add     ebx, ttychr
534      <1>      ; 23/07/2014 (always overwrite)
535      <1>      ;;cmp    word [ebx], 0
536      <1>      ;;ja     short com_eoi
537      <1>      ;
538 00011CA8 668903      <1>      mov     [ebx], ax ; Save ascii code
539      <1>      ; scan code = 0
540      <1>      com_eoi:
541      <1>      ;mov     al, 20h
542      <1>      ;out     20h, al ; end of interrupt
543      <1>      ;
544      <1>      ; 07/11/2015
545      <1>      ;pop     eax ; *
546 00011CAB A0[0A770100] <1>      mov     al, byte [ccomport] ; current COM port
547      <1>      ; al = tty number (8 or 9)
548 00011CB0 E80F000000 <1>      call    wakeup
549      <1>      com_i4ret:
550      <1>      ; 23/10/2015
551 00011CB5 5A      <1>      pop     edx ; **
552 00011CB6 59      <1>      pop     ecx ; ***
553      <1>      ; 18/11/2015
554      <1>      ;pop     eax ; ****
555      <1>      ;mov     cr3, eax
556      <1>      ;jmp     iiret
557 00011CB7 E928F1FEFF <1>      jmp     iiretp
558      <1>
559      <1>      ;iiretp: ; 01/09/2015
560      <1>      ; ; 28/08/2015
561      <1>      ; pop     eax ; (*) page directory
562      <1>      ; mov     cr3, eax
563      <1>      ;iiret:
564      <1>      ; ; 22/08/2014
565      <1>      ; mov     al, 20h ; END OF INTERRUPT COMMAND TO 8259
566      <1>      ; out     20h, al ; 8259 PORT
567      <1>      ;
568      <1>      ; pop     es
569      <1>      ; pop     ds
570      <1>      ; pop     edi
571      <1>      ; pop     esi
572      <1>      ; pop     ebx ; 29/08/2014
573      <1>      ; pop     eax
574      <1>      ; iretd
575      <1>
576      <1>      ; 21/11/2023
577      <1>      %if 0
578      <1>
579      <1>      sp_init:
580      <1>      ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
581      <1>      ; 07/11/2015
582      <1>      ; 29/10/2015
583      <1>      ; 26/10/2015
584      <1>      ; 23/10/2015
585      <1>      ; 29/06/2015
586      <1>      ; 14/03/2015 (Retro UNIX 386 v1 - 115200 baud)
587      <1>      ; 28/07/2014 (Retro UNIX 8086 v1 - 9600 baud)
588      <1>      ; Initialization of Serial Port Communication Parameters
589      <1>      ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
590      <1>      ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
591      <1>
592      <1>      ; ((Modified registers: EAX, ECX, EDX, EBX))
593      <1>
594      <1>      ; INPUT: (29/06/2015)
595      <1>      ; AL = 0 for COM1
596      <1>      ; 1 for COM2
597      <1>      ; AH = Communication parameters
598      <1>
599      <1>      ; (*) Communication parameters (except BAUD RATE):
600      <1>      ; Bit 4 3 2 1 0
601      <1>      ; -PARITY-- STOP BIT -WORD LENGTH-
602      <1>      ; this one --> 00 = none 0 = 1 bit 11 = 8 bits
603      <1>      ; 01 = odd 1 = 2 bits 10 = 7 bits
604      <1>      ; 11 = even
605      <1>      ; Baud rate setting bits: (29/06/2015)
606      <1>      ; Retro UNIX 386 v1 feature only !
607      <1>      ; Bit 7 6 5 | Baud rate
608      <1>      ; -----
609      <1>      ; value 0 0 0 | Default (Divisor = 1)
610      <1>      ; 0 0 1 | 9600 (12)
611      <1>      ; 0 1 0 | 19200 (6)
612      <1>      ; 0 1 1 | 38400 (3)
613      <1>      ; 1 0 0 | 14400 (8)
614      <1>      ; 1 0 1 | 28800 (4)
615      <1>      ; 1 1 0 | 57600 (2)
616      <1>      ; 1 1 1 | 115200 (1)
617      <1>
618      <1>      ; References:
619      <1>      ; (1) IBM PC-XT Model 286 BIOS Source Code
620      <1>      ; RS232.ASM --- 10/06/1985 COMMUNICATIONS BIOS (RS232)
621      <1>      ; (2) Award BIOS 1999 - ATORGS.ASM
622      <1>      ; (3) http://wiki.osdev.org/Serial_Ports
623      <1>
624      <1>      ; Set communication parameters for COM1 (= 03h)
625      <1>
626      <1>      mov     ebx, com1p ; COM1 parameters
627      <1>      mov     dx, 3F8h ; COM1
628      <1>      ; 29/10/2015
629      <1>      mov     cx, 301h ; divisor = 1 (115200 baud)
630      <1>      call    sp_i3 ; call A4
631      <1>      test     al, 80h
632      <1>      jz      short sp_i0 ; OK..
633      <1>      ; Error !
634      <1>      ;mov     dx, 3F8h
635      <1>      sub     dl, 5 ; 3FDh -> 3F8h
636      <1>      mov     cx, 30Eh ; divisor = 12 (9600 baud)
637      <1>      call    sp_i3 ; call A4
638      <1>      test     al, 80h
639      <1>      jnz     short sp_i1
640      <1>      sp_i0:
641      <1>      ; (Note: Serial port interrupts will be disabled here...)
642      <1>      ; (INT 14h initialization code disables interrupts.)
643      <1>
644      <1>      mov     byte [ebx], 0E3h ; 11100011b

```

```

645     call    sp_i5 ; 29/06/2015
646     <1> sp_i1:
647     <1> inc     ebx
648     <1> mov     dx, 2F8h          ; COM2
649     <1> ; 29/10/2015
650     <1> mov     cx, 301h ; divisor = 1 (115200 baud)
651     <1> call    sp_i3 ; call A4
652     <1> test    al, 80h
653     <1> jz      short sp_i2 ; OK..
654     <1> ; Error !
655     <1> ;mov     dx, 2F8h
656     <1> sub     dl, 5 ; 2FDh -> 2F8h
657     <1> mov     cx, 30Eh ; divisor = 12 (9600 baud)
658     <1> call    sp_i3 ; call A4
659     <1> test    al, 80h
660     <1> jnz     short sp_i7
661     <1> sp_i2:
662     <1> mov     byte [ebx], 0E3h ; 11100011b
663     <1> sp_i6:
664     <1> ; COM2 - enabling IRQ 3
665     <1> ; 29/07/2022
666     <1> ; 07/11/2015
667     <1> ; 26/10/2015
668     <1> pushf
669     <1> cli
670     <1> ;
671     <1> mov     dx, 2FCh          ; modem control register
672     <1> in      al, dx           ; read register
673     <1> JMP     $+2              ; I/O DELAY
674     <1> or      al, 8           ; enable bit 3 (OUT2)
675     <1> out     dx, al          ; write back to register
676     <1> JMP     $+2              ; I/O DELAY
677     <1> ;mov     dx, 2F9h          ; interrupt enable register
678     <1> ; 29/07/2022
679     <1> mov     dl, 0F9h
680     <1> in      al, dx           ; read register
681     <1> JMP     $+2              ; I/O DELAY
682     <1> ;or      al, 1           ; receiver data interrupt enable and
683     <1> or      al, 3           ; transmitter empty interrupt enable
684     <1> out     dx, al          ; write back to register
685     <1> JMP     $+2              ; I/O DELAY
686     <1> in      al, 21h         ; read interrupt mask register
687     <1> JMP     $+2              ; I/O DELAY
688     <1> and     al, 0F7h        ; enable IRQ 3 (COM2)
689     <1> out     21h, al         ; write back to register
690     <1> ;
691     <1> ; 23/10/2015
692     <1> mov     eax, com2_int
693     <1> mov     [com2_irq3], eax
694     <1> ; 26/10/2015
695     <1> popf
696     <1> sp_i7:
697     <1> retn
698
699     <1> sp_i3:
700     <1> ;A4: ;----- INITIALIZE THE COMMUNICATIONS PORT
701     <1> ; 28/10/2015
702     <1> inc     dl ; 3F9h (2F9h) ; 3F9h, COM1 Interrupt enable register
703     <1> mov     al, 0
704     <1> out     dx, al           ; disable serial port interrupt
705     <1> JMP     $+2              ; I/O DELAY
706     <1> add     dl, 2 ; 3FBh (2FBh) ; COM1 Line control register (3FBh)
707     <1> mov     al, 80h
708     <1> out     dx, al          ; SET DLAB=1 ; divisor latch access bit
709     <1> ;----- SET BAUD RATE DIVISOR
710     <1> ; 26/10/2015
711     <1> sub     dl, 3 ; 3F8h (2F8h) ; register for least significant byte
712     <1> ; of the divisor value
713     <1> mov     al, cl ; 1
714     <1> out     dx, al          ; 1 = 115200 baud (Retro UNIX 386 v1)
715     <1> ; 2 = 57600 baud
716     <1> ; 3 = 38400 baud
717     <1> ; 6 = 19200 baud
718     <1> ; 12 = 9600 baud (Retro UNIX 8086 v1)
719     <1> JMP     $+2              ; I/O DELAY
720     <1> sub     al, al
721     <1> inc     dl ; 3F9h (2F9h) ; register for most significant byte
722     <1> ; of the divisor value
723     <1> out     dx, al ; 0
724     <1> JMP     $+2              ; I/O DELAY
725     <1> ;
726     <1> mov     al, ch ; 3 ; 8 data bits, 1 stop bit, no parity
727     <1> ;and     al, 1Fh ; Bits 0,1,2,3,4
728     <1> add     dl, 2 ; 3FBh (2FBh) ; Line control register
729     <1> out     dx, al
730     <1> JMP     $+2              ; I/O DELAY
731     <1> ; 29/10/2015
732     <1> dec     dl ; 3FAh (2FAh) ; FIFO Control register (16550/16750)
733     <1> xor     al, al          ; 0
734     <1> out     dx, al          ; Disable FIFOs (reset to 8250 mode)
735     <1> JMP     $+2
736     <1> sp_i4:
737     <1> ;A18: ;----- COMM PORT STATUS ROUTINE
738     <1> ; 29/06/2015 (line status after modem status)
739     <1> add     dl, 4 ; 3FEh (2FEh) ; Modem status register
740     <1> sp_i4s:
741     <1> in      al, dx           ; GET MODEM CONTROL STATUS
742     <1> JMP     $+2              ; I/O DELAY
743     <1> mov     ah, al          ; PUT IN (AH) FOR RETURN
744     <1> dec     dl ; 3FDh (2FDh) ; POINT TO LINE STATUS REGISTER
745     <1> ; dx = 3FDh for COM1, 2FDh for COM2
746     <1> in      al, dx           ; GET LINE CONTROL STATUS
747     <1> ; AL = Line status, AH = Modem status
748     <1> retn
749
750     <1> sp_status:
751     <1> ; 29/06/2015
752     <1> ; 27/06/2015 (Retro UNIX 386 v1)
753     <1> ; Get serial port status
754     <1> mov     dx, 3FEh          ; Modem status register (COM1)
755     <1> sub     dh, al          ; dh = 2 for COM2 (al = 1)
756     <1> ; dx = 2FEh for COM2
757     <1> jmp     short sp_i4s
758
759     <1> sp_setp: ; Set serial port communication parameters
760     <1> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
761     <1> ; 07/11/2015
762     <1> ; 29/10/2015
763     <1> ; 29/06/2015
764     <1> ; Retro UNIX 386 v1 feature only !
765     <1> ;
766     <1> ; INPUT:
767     <1> ; AL = 0 for COM1
768     <1> ; 1 for COM2

```



```

769      ; AH = Communication parameters (*)
770      ; OUTPUT:
771      ; CL = Line status
772      ; CH = Modem status
773      ; If cf = 1 -> Error code in [u.error]
774      ; 'invalid parameter !'
775      ; or
776      ; 'device not ready !' error
777
778      ; (*) Communication parameters (except BAUD RATE):
779      ; Bit 4 3 2 1 0
780      ; -PARITY-- STOP BIT -WORD LENGTH-
781      ; this one --> 00 = none 0 = 1 bit 11 = 8 bits
782      ; 01 = odd 1 = 2 bits 10 = 7 bits
783      ; 11 = even
784      ; Baud rate setting bits: (29/06/2015)
785      ; Retro UNIX 386 v1 feature only !
786      ; Bit 7 6 5 | Baud rate
787      ; -----
788      ; value 0 0 0 | Default (Divisor = 1)
789      ; 0 0 1 | 9600 (12)
790      ; 0 1 0 | 19200 (6)
791      ; 0 1 1 | 38400 (3)
792      ; 1 0 0 | 14400 (8)
793      ; 1 0 1 | 28800 (4)
794      ; 1 1 0 | 57600 (2)
795      ; 1 1 1 | 115200 (1)
796
797      ; (COM1 base port address = 3F8h, COM1 Interrupt = IRQ 4)
798      ; (COM2 base port address = 2F8h, COM1 Interrupt = IRQ 3)
799
800      ; ((Modified registers: EAX, ECX, EDX, EBX))
801
802      mov dx, 3F8h
803      mov ebx, com1p ; COM1 control byte offset
804      cmp al, 1
805      ja short sp_invp_err
806      jb short sp_setp1 ; COM1 (AL = 0)
807      dec dh ; 2F8h
808      inc ebx ; COM2 control byte offset
809
810      sp_setp1:
811      ; 29/10/2015
812      mov [ebx], ah
813      movzx ecx, ah
814      shr cl, 5 ; -> baud rate index
815      and ah, 1Fh ; communication parameters except baud rate
816      mov al, [ecx+b_div_tbl]
817      mov cx, ax
818      call sp_i3
819      mov cx, ax ; CL = Line status, CH = Modem status
820      test al, 80h
821      jz short sp_setp2
822      mov byte [ebx], 0E3h ; Reset to initial value (11100011b)
823
824      stp_dnr_err:
825      mov dword [u.error], ERR_DEV_NOT_RDY ; 'device not ready !'
826      ; CL = Line status, CH = Modem status
827      stc
828      ret
829
830      sp_setp2:
831      cmp dh, 2 ; COM2 (2F?h)
832      jna sp_i6
833      ; COM1 (3F?h)
834      ; 29/07/2022
835      ja short sp_i5
836      jmp sp_i6
837
838      sp_i5:
839      ; 29/07/2022
840      ; 07/11/2015
841      ; 26/10/2015
842      ; 29/06/2015
843      ;
844      ; COM1 - enabling IRQ 4
845      pushf
846      cli
847      mov dx, 3FCh ; modem control register
848      in al, dx ; read register
849      JMP $+2 ; I/O DELAY
850      or al, 8 ; enable bit 3 (OUT2)
851      out dx, al ; write back to register
852      JMP $+2 ; I/O DELAY
853      mov dx, 3F9h ; interrupt enable register
854      ; 29/07/2022
855      mov dl, 0F9h
856      in al, dx ; read register
857      JMP $+2 ; I/O DELAY
858      or al, 1 ; receiver data interrupt enable and
859      or al, 3 ; transmitter empty interrupt enable
860      out dx, al ; write back to register
861      JMP $+2 ; I/O DELAY
862      in al, 21h ; read interrupt mask register
863      JMP $+2 ; I/O DELAY
864      and al, 0EFh ; enable IRQ 4 (COM1)
865      out 21h, al ; write back to register
866      ;
867      ; 23/10/2015
868      mov eax, com1_int
869      mov [com1_irq4], eax
870      ; 26/10/2015
871      popf
872      ret
873
874      sp_invp_err:
875      mov dword [u.error], ERR_INV_PARAMETER ; 'invalid parameter !'
876      xor ecx, ecx
877      dec ecx ; 0FFFFh
878      stc
879      ret
880
881      ; 29/10/2015
882      b_div_tbl: ; Baud rate divisor table (115200/divisor)
883      db 1, 12, 6, 3, 8, 4, 1
884
885      %endif
886
887      ; 23/10/2015
888      com1_irq4:
889      dd dummy_retn
890      com2_irq3:
891      dd dummy_retn
892
893      ; 21/11/2023
894      dummy_retn:
895      ret
896      wakeup:

```

```

893      <1>      ; 24/01/2016
894 00011CC4 C3      <1>      retn
895      <1>
896      <1> set_working_path_x:
897      <1>      ; 17/10/2016 (TRDOS 386 - FFF & FNF)
898      <1>      ;mov     ax, 1
899      <1>      ; File name is needed/forced (AL=1)
900      <1>      ; Change directory as temporary (AH=0)
901      <1>      ; 29/07/2022
902 00011CC5 31C0      <1>      xor     eax, eax
903 00011CC7 FEC0      <1>      inc     al
904      <1>      ; eax = 1
905      <1> set_working_path_xx: ; 30/12/2017 (syschdir)
906      <1>      ; This is needed for preventing wrong Find Next File
907      <1>      ; system call after sysopen, syscreate, sysmkdir etc.
908      <1>      ; Find Next File must immediate follow Find First File)
909      <1>
910 00011CC9 8825[58830100] <1>      mov     [FFF_valid], ah ; 0 ; reset ; 17/10/2016
911      <1>
912      <1> set_working_path:
913      <1>      ; 08/08/2022
914      <1>      ; 29/07/2022 - TRDOS 386 Kernel v2.0.5
915      <1>      ; 16/10/2016
916      <1>      ; 12/10/2016
917      <1>      ; 10/10/2016
918      <1>      ; 05/10/2016 - TRDOS 386 (TRDOS v2.0)
919      <1>
920      <1>      ; TRDOS v1.0 (DIR.ASM, "proc_set_working_path")
921      <1>      ; 27/01/2011 - 08/02/2011
922      <1>      ; Set/Changes current drive, directory and file
923      <1>      ; depending on command tail
924      <1>      ; (procedure is derivated from CMD_INTR.ASM
925      <1>      ; file or dir locating code of internal commands)
926      <1>      ; (This procedure is prepared for INT 21H file/dir
927      <1>      ; functions and also to get compact code for
928      <1>      ; internal mainprog -command interpreter- commands)
929      <1>
930      <1>      ; INPUT: DS:SI -> Command tail (ASCIIIZ string)
931      <1>      ; AL = 0 -> any, AL > 0 -> file name is forced
932      <1>      ; AH = CD -> change directory permanently
933      <1>      ; AH <> CD -> Change directory as temporary
934      <1>
935      <1>      ; OUTPUT: ES=DS, FindFile structure has been set
936      <1>      ; RUN_CDRV points previous current drive
937      <1>      ; DS:SI = FindFile structure address
938      <1>      ; (DS=CS)
939      <1>      ; AX, BX, CX, DX, DI will be changed
940      <1>      ; cf = 1 -> Error code in AX (AL)
941      <1>      ; stc & AX = 0 -> Bad command or path name
942      <1>      -----
943      <1>
944      <1>      ; TRDOS 386 (05/10/2016)
945      <1>      ; INPUT:
946      <1>      ; ESI = File/Directory Path (ASCIIIZ string)
947      <1>      ; address in user's memory space
948      <1>      ; AL = 0 -> any
949      <1>      ; AL > 0 -> file name is forced
950      <1>      ; AH = CD -> change directory as permanent
951      <1>      ; AH <> CD -> change directory as temporary
952      <1>
953      <1>      ; OUTPUT:
954      <1>      ; FindFile structure has been set
955      <1>      ; RUN_CDRV points previous current drive
956      <1>      ; ESI = FindFile_Name address ; 12/10/2016
957      <1>
958      <1>      ; cf = 1 -> Error code in EAX (AL)
959      <1>      ; stc & EAX = 0 -> Bad command or path name
960      <1>
961      <1>      ; Modified registers: EAX, EBX, ECX, EDX, ESI, EDI
962      <1>
963 00011CCF 66A3[5C830100] <1>      mov     [SWP_Mode], ax
964 00011CD5 A0[62770100] <1>      mov     al, [Current_Drv]
965 00011CDA 30E4      <1>      xor     ah, ah
966 00011CDC 66A3[5E830100] <1>      mov     [SWP_DRV], ax
967      <1>
968      <1>      ; TRDOS 386 ring 3 (user's page directory)
969      <1>      ; to ring 0 (kernel's page directory)
970      <1>      ; transfer modifications (05/10/2016).
971      <1>
972 00011CE2 55      <1>      push    ebp
973 00011CE3 89E5      <1>      mov     ebp, esp
974      <1>
975      <1>      ;mov     ecx, 128 ; maximum path length = 128 bytes
976      <1>      ; 29/07/2022
977 00011CE5 31C9      <1>      xor     ecx, ecx
978 00011CE7 8180      <1>      mov     cl, 128
979 00011CE9 29CC      <1>      sub     esp, ecx ; reserve 128 bytes (buffer) on stack
980 00011CEB 89E7      <1>      mov     edi, esp ; destination address (kernel space)
981      <1>      ; esi = source address (virtual, in user's memory space)
982 00011CED E863EEFFFF <1>      call    transfer_from_user_buffer
983 00011CF2 720D      <1>      jc     short loc_swp_xor_retn
984      <1>
985 00011CF4 89E6      <1>      mov     esi, esp ; temporary buffer (the path) on stack
986      <1> loc_swp_fchar:
987 00011CF6 8A06      <1>      mov     al, [esi]
988 00011CF8 3C20      <1>      cmp     al, 20h
989 00011CFA 7711      <1>      ja     short loc_swp_parse_path_name
990      <1>      ;je     short loc_swp_fchar_next
991      <1>      ; 29/07/2022
992 00011CFC 7203      <1>      jb     short loc_swp_xor_retn
993      <1>
994      <1> loc_swp_fchar_next:
995 00011CFE 46      <1>      inc     esi
996 00011CFF EBF5      <1>      jmp     short loc_swp_fchar
997      <1>
998      <1> loc_swp_xor_retn:
999 00011D01 31C0      <1>      xor     eax, eax
1000 00011D03 F9      <1>      stc
1001      <1> loc_swp_retn:
1002 00011D04 89EC      <1>      mov     esp, ebp
1003 00011D06 5D      <1>      pop     ebp
1004      <1>
1005      <1>      ;mov     esi, FindFile_Drv
1006 00011D07 BE[44800100] <1>      mov     esi, FindFile_Name ; 12/10/2016
1007 00011D0C C3      <1>      retn
1008      <1>
1009      <1> ;loc_swp_fchar_next:
1010      <1> ; inc     esi
1011      <1> ; jmp     short loc_swp_fchar
1012      <1>
1013      <1> loc_swp_parse_path_name:
1014 00011D0D 8F[02800100] <1>      mov     edi, FindFile_Drv
1015 00011D12 E8578CFFFF <1>      call    parse_path_name
1016 00011D17 72EB      <1>      jc     short loc_swp_retn

```

```

1017 <1>
1018 <1> loc_swap_checkfile_name:
1019 00011D19 803D[5C830100]00 <1> cmp byte [SWP_Mode], 0
1020 00011D20 761E <1> jna short loc_swap_drv
1021 <1>
1022 <1> ; 10/10/2016 (valid file name checking)
1023 00011D22 BE[44800100] <1> mov esi, FindFile_Name
1024 00011D27 803E20 <1> cmp byte [esi], 20h
1025 00011D2A 76D5 <1> jna short loc_swap_xor_retn
1026 <1>
1027 <1> ; 16/10/2016
1028 00011D2C C605[5B830100]00 <1> mov byte [SWP_inv_fname], 0 ; reset
1029 <1> ; esi = file name address (ASCIIIZ)
1030 00011D33 E8916FFFFF <1> call check_filename
1031 00011D38 7306 <1> jnc short loc_swap_drv
1032 <1>
1033 00011D3A FE05[5B830100] <1> inc byte [SWP_inv_fname] ; set
1034 <1> loc_swap_drv:
1035 00011D40 8A35[62770100] <1> mov dh, [Current_Drv]
1036 <1> ;mov [RUN_CDRV], dh
1037 <1>
1038 00011D46 8A15[02800100] <1> mov dl, [FindFile_Drv]
1039 <1> ;cmp dl, dh
1040 00011D4C 3A15[62770100] <1> cmp dl, [Current_Drv]
1041 00011D52 740D <1> je short loc_swap_change_directory
1042 <1>
1043 00011D54 FE05[5F830100] <1> inc byte [SWP_DRV_chg]
1044 00011D5A E88159FFFF <1> call change_current_drive
1045 00011D5F 72A3 <1> jc short loc_swap_retn ; eax = error code
1046 <1> ; eax = 0
1047 <1>
1048 <1> loc_swap_change_directory:
1049 00011D61 803D[03800100]21 <1> cmp byte [FindFile_Directory], 21h
1050 00011D68 F5 <1> cmc
1051 00011D69 7399 <1> jnc short loc_swap_retn
1052 <1>
1053 00011D6B FE05[5F830100] <1> inc byte [SWP_DRV_chg]
1054 00011D71 FE05[7E2E0100] <1> inc byte [Restore_CDIR]
1055 00011D77 BE[03800100] <1> mov esi, FindFile_Directory
1056 00011D7C 8A25[5D830100] <1> mov ah, [SWP_Mode+1]
1057 00011D82 E85486FFFF <1> call change_current_directory
1058 <1> ;jc short loc_swap_retn ; eax = error code
1059 <1> ; 08/08/2022
1060 00011D87 7305 <1> jnc short loc_swap_change_prompt_dir_string
1061 00011D89 E976FFFFFF <1> jmp loc_swap_retn
1062 <1>
1063 <1> loc_swap_change_prompt_dir_string:
1064 <1> ; esi = PATH_Array
1065 <1> ; eax = Current Directory First Cluster
1066 <1> ; edi = Logical DOS Drive Description Table
1067 00011D8E E87685FFFF <1> call change_prompt_dir_str
1068 00011D93 29C0 <1> sub eax, eax ; 0
1069 00011D95 E96AFFFFFF <1> jmp loc_swap_retn
1070 <1>
1071 <1> reset_working_path:
1072 <1> ; 06/10/2016 - TRDOS 386 (TRDOS v2.0)
1073 <1> ;
1074 <1> ; TRDOS v1.0 (DIR.ASM, "proc_reset_working_path")
1075 <1> ; 05/02/2011 - 08/02/2011
1076 <1> ;
1077 <1> ; Restores current drive and directory
1078 <1> ;
1079 <1> ; INPUT: none
1080 <1> ; OUTPUT: DL = SWP_DRV, EAX = 0 -> OK
1081 <1> ;
1082 <1> ; AX = 0 -> ESI = Logical Dos Drv Desc. Table
1083 <1> ;
1084 <1> ; EAX, EBX, ECX, EDX, ESI, EDI will be changed
1085 <1> ;
1086 <1> ;
1087 <1>
1088 00011D9A 31C0 <1> xor eax, eax
1089 00011D9C 48 <1> dec eax
1090 <1>
1091 00011D9D 668B15[5E830100] <1> mov dx, [SWP_DRV]
1092 00011DA4 08F6 <1> or dh, dh
1093 00011DA6 742E <1> jz short loc_rwp_return
1094 <1>
1095 00011DA8 3A15[62770100] <1> cmp dl, [Current_Drv]
1096 00011DAE 7407 <1> je short loc_rwp_restore_cdir
1097 <1> loc_rwp_restore_cdrv:
1098 00011DB0 E82B59FFFF <1> call change_current_drive
1099 00011DB5 EB10 <1> jmp short loc_rwp_restore_ok
1100 <1> loc_rwp_restore_cdir:
1101 00011DB7 31DB <1> xor ebx, ebx
1102 00011DB9 88D7 <1> mov bh, dl
1103 00011DBB BE00010900 <1> mov esi, Logical_DOSDisks
1104 00011DC0 01DE <1> add esi, ebx
1105 <1>
1106 00011DC2 E8CA59FFFF <1> call restore_current_directory
1107 <1>
1108 <1> loc_rwp_restore_ok:
1109 00011DC7 668B15[5E830100] <1> mov dx, [SWP_DRV]
1110 00011DCE 31C0 <1> xor eax, eax
1111 00011DD0 66A3[5F830100] <1> mov [SWP_DRV_chg], ax
1112 <1> loc_rwp_return:
1113 00011DD6 C3 <1> retn
1114 <1>
1115 <1> get_file_name:
1116 <1> ; 25/08/2024 (TRDOS 386 Kernel v2.0.9)
1117 <1> ; 29/07/2022 (TRDOS 386 Kernel v2.0.5)
1118 <1> ; 15/10/2016 - TRDOS 386 (TRDOS v2.0)
1119 <1> ; Convert file name
1120 <1> ; from directory entry format
1121 <1> ; to (8.3) dot file name format
1122 <1> ;
1123 <1> ; TRDOS v1.0 (DIR.ASM, "get_file_name")
1124 <1> ; 2005 - 09/10/2011
1125 <1> ; INPUT:
1126 <1> ; DS:SI -> Directory Entry Format File Name
1127 <1> ; ES:DI -> DOS Dot File Name Address
1128 <1> ; OUTPUT:
1129 <1> ; DS:SI -> DOS Dot File Name Address
1130 <1> ; ES:DI -> Directory Entry Format File Name
1131 <1> ;
1132 <1> ; TRDOS 386 (15/10/2016)
1133 <1> ; INPUT:
1134 <1> ; ESI = File name addr in dir entry format
1135 <1> ; EDI = Dot file name address (destination)
1136 <1> ; OUTPUT:
1137 <1> ; File name is converted and moved
1138 <1> ; to destination (as 8.3 dot filename)
1139 <1> ;
1140 <1> ; Modified registers: EAX, ECX

```

```

1141 <1>
1142 <1>
1143 <1> ; 2005 (TRDOS 8086) - 2016 (TRDOS 386)
1144 00011DD7 57 <1> push edi
1145 00011DD8 56 <1> push esi
1146 00011DD9 AC <1> lodsb
1147 <1> ; 25/08/2024
1148 00011DDA 31C9 <1> xor ecx, ecx ; 0
1149 00011DDC 3C20 <1> cmp al, 20h
1150 00011DDE 7626 <1> jna short pass_gfn_ext
1151 <1> ; 25/08/2024
1152 <1> ; push esi
1153 00011DE0 AA <1> stosb
1154 <1> ; 25/08/2024
1155 <1> ; 29/07/2022
1156 <1> ; xor ecx, ecx
1157 <1> ; ecx <= 128 ; 25/08/2024
1158 00011DE1 B107 <1> mov cl, 7
1159 <1> ; 25/08/2024
1160 00011DE3 01CE <1> add esi, ecx ; add esi, 7
1161 00011DE5 56 <1> push esi ; (*)
1162 <1> loc_gfn_next_char:
1163 00011DE6 AC <1> lodsb
1164 00011DE7 3C20 <1> cmp al, 20h
1165 00011DE9 7603 <1> jna short pass_gfn_fn
1166 00011DEB AA <1> stosb
1167 00011DEC E2F8 <1> loop loc_gfn_next_char
1168 <1> pass_gfn_fn:
1169 <1> ; pop esi
1170 <1> ; add esi, 7
1171 <1> ; 25/08/2024
1172 00011DEE 5E <1> pop esi ; (*)
1173 <1>
1174 00011DEF AC <1> lodsb
1175 00011DF0 3C20 <1> cmp al, 20h
1176 00011DF2 7612 <1> jna short pass_gfn_ext
1177 00011DF4 B42E <1> mov ah, 1
1178 00011DF6 86C4 <1> xchg ah, al
1179 00011DF8 66AB <1> stosw
1180 00011DFA AC <1> lodsb
1181 00011DFB 3C20 <1> cmp al, 20h
1182 00011DFD 7607 <1> jna short pass_gfn_ext
1183 00011DFF AA <1> stosb
1184 00011E00 AC <1> lodsb
1185 00011E01 3C20 <1> cmp al, 20h
1186 00011E03 7601 <1> jna short pass_gfn_ext
1187 00011E05 AA <1> stosb
1188 <1> pass_gfn_ext:
1189 00011E06 30C0 <1> xor al, al
1190 00011E08 AA <1> stosb
1191 00011E09 5E <1> pop esi
1192 00011E0A 5F <1> pop edi
1193 <1> ; 25/08/2024
1194 <1> ; ecx <= 7
1195 00011E0B C3 <1> retn
1196 <1>
1197 <1> set_hardware_int_vector:
1198 <1> ; 18/03/2017
1199 <1> ; 03/03/2017
1200 <1> ; 28/02/2017 - TRDOS 386 (TRDOS v2.0)
1201 <1> ;
1202 <1> ; SET/RESET HARDWARE INTERRUPT GATE
1203 <1> ;
1204 <1> ; Changes interrupt gate descriptor table
1205 <1> ; (without changing default interrupt list)
1206 <1> ;
1207 <1> ; INPUT:
1208 <1> ; AL = IRQ number (0 to 15)
1209 <1> ; AH > 0 -> set
1210 <1> ; AH = 0 -> reset
1211 <1> ;
1212 <1> ; Modified registers: eax, ebx, edx, edi
1213 <1> ;
1214 <1>
1215 00011E0C C0E002 <1> shl al, 2 ; IRQ number * 4
1216 00011E0F 0FB6D8 <1> movzx ebx, al
1217 <1>
1218 00011E12 08E4 <1> or ah, ah
1219 00011E14 7508 <1> jnz short shintv_1 ; set (for user call service)
1220 <1>
1221 <1> ; 18/03/2017
1222 00011E16 81C3[68360100] <1> add ebx, IRQ_list ; reset to default interrupt list
1223 00011E1C EB06 <1> jmp short shintv_2
1224 <1> shintv_1:
1225 00011E1E 81C3[451E0100] <1> add ebx, IRQ_u_list
1226 <1> shintv_2:
1227 00011E24 8B13 <1> mov edx, [ebx] ; IRQ handler address
1228 <1>
1229 <1> ; 03/03/2017
1230 00011E26 D0E0 <1> shl al, 1 ; IRQ number * 8
1231 <1> ; 18/03/2017
1232 00011E28 0FB6F8 <1> movzx edi, al
1233 00011E2B 81C7[B8740100] <1> add edi, idt + (8*32) ; IRQ 0 offset = idt + 256
1234 <1>
1235 00011E31 89D0 <1> mov eax, edx ; IRQ handler address
1236 00011E33 BB00000800 <1> mov ebx, 80000h
1237 <1>
1238 <1> ; mov
1239 00011E38 66BA008E <1> mov dx, 8E00h
1240 00011E3C 6689C3 <1> mov bx, ax
1241 00011E3F 89D8 <1> mov eax, ebx ; /* selector = 0x0008 = cs */
1242 <1> ; /* interrupt gate - dpl=0, present */
1243 00011E41 AB <1> stosd ; selector & offset bits 0-15
1244 00011E42 8917 <1> mov [edi], edx ; attributes & offset bits 16-23
1245 <1>
1246 00011E44 C3 <1> retn
1247 <1> IRQ_u_list:
1248 <1> ; 28/02/2017
1249 00011E45 [88090000] <1> dd timer_int
1250 00011E49 [ED100000] <1> dd kb_int
1251 00011E4D [740B0000] <1> dd irq2
1252 00011E51 [851E0100] <1> dd IRQ_service3
1253 00011E55 [8F1E0100] <1> dd IRQ_service4
1254 00011E59 [991E0100] <1> dd IRQ_service5
1255 00011E5D [A04E0000] <1> dd fdc_int
1256 00011E61 [A31E0100] <1> dd IRQ_service7
1257 00011E65 [FC0A0000] <1> dd rtc_int
1258 00011E69 [AD1E0100] <1> dd IRQ_service9
1259 00011E6D [B71E0100] <1> dd IRQ_service10
1260 00011E71 [C11E0100] <1> dd IRQ_service11
1261 00011E75 [CB1E0100] <1> dd IRQ_service12
1262 00011E79 [D51E0100] <1> dd IRQ_service13
1263 00011E7D [DF570000] <1> dd hdc1_int
1264 00011E81 [02580000] <1> dd hdc2_int

```

```

1265 <1>
1266 <1> ; 03/03/2017
1267 <1> ; 27/02/2017
1268 <1> IRQ_service3:
1269 00011E85 36C605[cc880100]03 <1> mov byte [ss:IRQnum], 3
1270 00011E8D EB4E <1> jmp short IRQ_service
1271 <1> IRQ_service4:
1272 00011E8F 36C605[cc880100]04 <1> mov byte [ss:IRQnum], 4
1273 00011E97 EB44 <1> jmp short IRQ_service
1274 <1> IRQ_service5:
1275 00011E99 36C605[cc880100]05 <1> mov byte [ss:IRQnum], 5
1276 00011EA1 EB3A <1> jmp short IRQ_service
1277 <1> IRQ_service7:
1278 00011EA3 36C605[cc880100]07 <1> mov byte [ss:IRQnum], 7
1279 00011EAB EB30 <1> jmp short IRQ_service
1280 <1> IRQ_service9:
1281 00011EAD 36C605[cc880100]09 <1> mov byte [ss:IRQnum], 9
1282 00011EB5 EB26 <1> jmp short IRQ_service
1283 <1> IRQ_service10:
1284 00011EB7 36C605[cc880100]0A <1> mov byte [ss:IRQnum], 10
1285 00011EBF EB1C <1> jmp short IRQ_service
1286 <1> IRQ_service11:
1287 00011EC1 36C605[cc880100]0B <1> mov byte [ss:IRQnum], 11
1288 00011EC9 EB12 <1> jmp short IRQ_service
1289 <1> IRQ_service12:
1290 00011ECB 36C605[cc880100]0C <1> mov byte [ss:IRQnum], 12
1291 00011ED3 EB08 <1> jmp short IRQ_service
1292 <1> IRQ_service13:
1293 00011ED5 36C605[cc880100]0D <1> mov byte [ss:IRQnum], 13
1294 <1> ; jmp short IRQ_service
1295 <1> IRQ_service:
1296 <1> ; 29/07/2022 (TRDOS 386 kernel v2.0.5)
1297 <1> ; 13/06/2017
1298 <1> ; 11/06/2017
1299 <1> ; 10/06/2017
1300 <1> ; 01/03/2017, 04/03/2017
1301 <1> ; 27/02/2017, 28/02/2017
1302 00011EDD 1E <1> push ds
1303 00011EDE 06 <1> push es
1304 00011EDF 0FA0 <1> push fs
1305 00011EE1 0FA8 <1> push gs
1306 <1>
1307 00011EE3 60 <1> pushad ; eax,ecx,edx,ebx,esp,ebp,esi,edi
1308 00011EE4 66B91000 <1> mov cx, KDATA
1309 00011EE8 8ED9 <1> mov ds, cx
1310 00011EEA 8EC1 <1> mov es, cx
1311 00011EEC 8EE1 <1> mov fs, cx
1312 00011EEE 8EE9 <1> mov gs, cx
1313 <1>
1314 00011EF0 0F20D8 <1> mov eax, cr3
1315 00011EF3 A3[C8880100] <1> mov [IRQ_cr3], eax
1316 <1>
1317 00011EF8 A1[A0760100] <1> mov eax, [k_page_dir]
1318 00011EFD 0F22D8 <1> mov cr3, eax
1319 <1>
1320 00011F00 A0[CC880100] <1> mov al, [IRQnum]
1321 <1>
1322 <1> ; mov cl, [sysflg]
1323 <1> ; mov [u.r_mode], cl ; system (0) or user mode (FFh)
1324 <1> IRQsrv_0:
1325 00011F05 0FB6D8 <1> movzx ebx, al
1326 00011F08 8A9B[A0350100] <1> mov bl, [ebx+IRQenum] ; IRQ (available) index number + 1
1327 <1> ; 01/03/2017
1328 00011F0E FECB <1> dec bl ; IRQ index number, 0 to 8
1329 <1> ; js IRQsrv_5 ; not available to use here!
1330 <1> ; 29/07/2022
1331 00011F10 785E <1> js short IRQsrv_j5 ; (jump to IRQsrv_5)
1332 <1>
1333 00011F12 80BB[92880100]80 <1> cmp byte [ebx+IRQ.method], 80h ; using by a dev or kernel?
1334 00011F19 7205 <1> jb short IRQsrv_1 ; no
1335 <1>
1336 <1> ; If the IRQ service is already owned by TRDOS 386 kernel
1337 <1> ; or a Device driver
1338 <1> ; we need to call 'dev_IRQ_service'
1339 <1>
1340 <1> ; IRQ number in AL
1341 00011F1B E80E010000 <1> call dev_IRQ_service ; IRQ service for device drivers
1342 <1> ; IRQ number in AL
1343 <1> IRQsrv_1:
1344 <1> ; check user callback service status
1345 <1> ; AL = IRQ number
1346 <1> ; EBX = IRQ (Available) Index number
1347 <1>
1348 00011F20 A2[CF8E0100] <1> mov [u.irqwait], al ; set waiting IRQ flag
1349 <1>
1350 00011F25 8A83[80880100] <1> mov al, [ebx+IRQ.owner]
1351 00011F2B 20C0 <1> and al, al
1352 <1> ; jz IRQsrv_5 ; it is not owned by a user/proc
1353 <1> ; 29/07/2022
1354 00011F2D 7441 <1> jz short IRQsrv_j5 ; (jump to IRQsrv_5)
1355 <1>
1356 <1> ; 03/03/2017
1357 00011F2F 89DA <1> mov edx, ebx
1358 00011F31 C0E202 <1> shl dl, 2
1359 00011F34 8B92[A4880100] <1> mov edx, [edx+IRQ.addr] ; S.R.B. or Callback service addr
1360 <1>
1361 00011F3A 8AA3[92880100] <1> mov ah, [ebx+IRQ.method]
1362 00011F40 F6C401 <1> test ah, 1
1363 00011F43 7530 <1> jnz short IRQsrv_4 ; Callback service method
1364 <1>
1365 <1> ; Signal Response Byte method
1366 <1> ; mov edx, [edx+IRQ.addr] ; Signal Response Byte address
1367 <1> ; ; (Physical address, non-swappable)
1368 00011F45 80E402 <1> and ah, 2 ; bit 1, (S.R.B.) counter (auto increment) method
1369 00011F48 8AA3[9B880100] <1> mov ah, [ebx+IRQ.srb] ; Signal Response Byte value
1370 00011F4E 7408 <1> jz short IRQsrv_2 ; fixed S.R.B. value
1371 <1> ; counter method (auto increment)
1372 00011F50 FEC4 <1> inc ah
1373 00011F52 88A3[9B880100] <1> mov [ebx+IRQ.srb], ah ; next (count) number
1374 <1> IRQsrv_2:
1375 00011F58 8822 <1> mov [edx], ah ; put S.R.B. val to the user's S.R.B. addr
1376 00011F5A C605[CF8E0100]00 <1> mov byte [u.irqwait], 0 ; clear waiting IRQ flag
1377 <1>
1378 00011F61 3A05[A58E0100] <1> cmp al, [u.uno]
1379 <1> ; je IRQsrv_5 ; the owner is current user/process
1380 <1> ; 29/07/2022
1381 00011F67 7407 <1> je short IRQsrv_j5 ; (jump to IRQsrv_5)
1382 <1> IRQsrv_3:
1383 <1> ; the owner is not current user/process
1384 <1> ; AL = process number
1385 00011F69 B202 <1> mov dl, 2 ; priority, 2 = event (high)
1386 00011F6B E87BFBFFFF <1> call set_run_sequence
1387 <1>
1388 <1> ; [u.irqwait] = waiting IRQ number for callback service

```

```

1389          <1> IRQsrv_j5:          ; 29/07/2022
1390 00011F70 E998000000          <1> jmp      IRQsrv_5
1391          <1> IRQsrv_4:
1392 00011F75 3A05[A58E0100]      <1> cmp     al, [u.uno] ; is the owner is current user/process?
1393 00011F7B 75EC                <1> jne     short IRQsrv_3 ; no !
1394          <1>
1395          <1> ; check if an IRQ callback service already in progress
1396 00011F7D 803D[D08E0100]00    <1> cmp     byte [u.r_lock], 0
1397          <1> ;ja      IRQsrv_5 ; nothing to do !
1398          <1> ; (we need to complete prev callback)
1399          <1> ; 29/07/2022
1400 00011F84 77EA                <1> ja      short IRQsrv_j5 ; (jump to IRQsrv_5)
1401          <1>
1402 00011F86 803D[CC8E0100]00    <1> cmp     byte [u.t_lock], 0
1403 00011F8D 777E                <1> ja      short IRQsrv_5 ; nothing to do !
1404          <1> ; (we need to complete timer callback)
1405          <1>
1406          <1> ; 04/03/2017
1407 00011F8F C605[CF8E0100]00    <1> mov     byte [u.irqwait], 0 ; reset/clear waiting IRQ flag
1408          <1>
1409 00011F96 FE05[D08E0100]      <1> inc     byte [u.r_lock] ; 'IRQ callback service in progress' flag
1410          <1>
1411 00011F9C 8A0D[488E0100]      <1> mov     cl, [sysflg] ; (system call) mode flag (kernel/user)
1412 00011FA2 880D[D18E0100]      <1> mov     [u.r_mode], cl ; system mode (0) or user mode (FFh)
1413          <1>
1414          <1> ;
1415 00011FA8 8B2D[3C760100]      <1> mov     ebp, [tss.esp0] ; kernel stack address (for ring 0)
1416 00011FAE 83ED14              <1> sub     ebp, 20 ; eip, cs, eflags, esp, ss
1417 00011FB1 892D[4C8E0100]      <1> mov     [u.sp], ebp
1418 00011FB7 8925[508E0100]      <1> mov     [u.usp], esp
1419          <1>
1420          <1> ;or      word [ebp+8], 200h ; 22/01/2017, force enabling interrupts
1421          <1>
1422 00011FBD 8B44241C            <1> mov     eax, [esp+28] ; pushed eax
1423 00011FC1 A3[548E0100]        <1> mov     [u.r0], eax
1424          <1>
1425 00011FC6 E8FCE9FFFF          <1> call    wswap ; save user's registers & status
1426          <1>
1427          <1> ; software int is in ring 0 but IRQ handler must return to ring 3
1428          <1> ; so, ring 3 return address and stack registers
1429          <1> ; (eip, cs, eflags, esp, ss)
1430          <1> ; must be copied to IRQ handler return
1431          <1> ; eip will be replaced by callback service routine address
1432          <1>
1433 00011FCB C605[488E0100]FF      <1> mov     byte [sysflg], 0FFh ; user mode
1434          <1>
1435          <1> ; system mode (system call)
1436          <1> ;mov     ebp, [u.sp] ; EIP (u), CS (UCODE), EFLAGS (u),
1437          <1> ; ESP (u), SS (UDATA)
1438          <1>
1439 00011FD2 8B4510              <1> mov     eax, [ebp+16] ; SS (UDATA)
1440 00011FD5 89E6                <1> mov     esi, esp
1441 00011FD7 50                  <1> push    eax
1442 00011FD8 50                  <1> push    eax
1443 00011FD9 89E7                <1> mov     edi, esp
1444 00011FDB 893D[508E0100]      <1> mov     [u.usp], edi
1445 00011FE1 B908000000          <1> mov     ecx, ((ESPACE/4) - 4) ; except DS, ES, FS, GS
1446 00011FE6 F3A5                <1> rep     movsd
1447 00011FE8 B104                <1> mov     cl, 4
1448 00011FEA F3AB                <1> rep     stosd
1449 00011FEC 893D[4C8E0100]      <1> mov     [u.sp], edi
1450 00011FF2 89EE                <1> mov     esi, ebp
1451 00011FF4 B105                <1> mov     cl, 5 ; EIP (u), CS (UCODE), EFLAGS (u), ESP (u), SS (UDATA)
1452 00011FF6 F3A5                <1> rep     movsd
1453          <1> ;
1454          <1>
1455 00011FF8 8B0D[AC8E0100]      <1> mov     ecx, [u.pgdir]
1456 00011FFE 890D[C8880100]      <1> mov     [IRQ_cr3], ecx
1457          <1>
1458          <1> set_IRQ_callback_addr:
1459          <1> ;
1460          <1> ; This routine sets return address
1461          <1> ; to start of user's interrupt
1462          <1> ; service (callback) address
1463          <1> ;
1464          <1> ; INPUT:
1465          <1> ; EDX = callback routine/service address
1466          <1> ; (virtual, not physical address!)
1467          <1> ; [u.sp] = kernel stack, points to
1468          <1> ; user's EIP,CS,EFLAGS,ESP,SS
1469          <1> ; registers.
1470          <1> ; OUTPUT:
1471          <1> ; EIP (user) = callback (service) address
1472          <1> ; CS (user) = UCODE
1473          <1> ; EFLAGS (user) = flags before callback
1474          <1> ; ESP (user) = ESP-4 (user, before callback)
1475          <1> ; [ESP](user) = EIP (user) before callback
1476          <1> ;
1477          <1> ; Note: If CPU was in user mode while entering
1478          <1> ; the timer interrupt service routine,
1479          <1> ; 'IRET' will get return to callback routine
1480          <1> ; immediately. If CPU was in system/kernel mode
1481          <1> ; 'iret' will get return to system call and
1482          <1> ; then, callback routine will be return address
1483          <1> ; from system call. (User's callback/service code
1484          <1> ; will be able to return to normal return address
1485          <1> ; via a 'sysrele' system call at the end.)
1486          <1> ;
1487          <1> ; Note: User's IRQ callback service code must be ended
1488          <1> ; with a 'sysrele' system call !
1489          <1> ;
1490          <1> ; For example:
1491          <1> ;
1492          <1> ; audio_IRQ_callback:
1493          <1> ; ...
1494          <1> ; <load DMA buffer with audio data>
1495          <1> ; ...
1496          <1> ; mov     eax, 39 ; 'sysrele'
1497          <1> ; int     40h ; TRDOS 386 system call (interrupt)
1498          <1> ;
1499          <1> ;
1500          <1> ;mov     edx, [edx+IRQ.addr] ; callback service address
1501          <1> ; (virtual address)
1502          <1>
1503 00012004 8B2D[4C8E0100]      <1> mov     ebp, [u.sp] ; kernel's stack, points to EIP (user)
1504 0001200A 895500              <1> mov     [ebp], edx
1505          <1> IRQsrv_5:
1506          <1> ; EOI & return
1507          <1> ; 01/08/2020
1508          <1> ; 11/06/2017
1509          <1> ; 10/06/2017
1510          <1> ;mov     al, [IRQnum]
1511 0001200D B020                <1> mov     al, 20h ; 01/08/2020
1512 0001200F FA                  <1> cli

```

```

1513                                     <1>         ;cmp     al, 7
1514 00012010 803D[cc880100]07          <1>         cmp     byte [IRQnum], 7 ; 01/08/2020
1515 00012017 7602                      <1>         jna     short IRQsrv_6
1516                                     <1>         ;
1517                                     <1>         ;;mov    al, EOI ; end of interrupt
1518                                     <1>         ;mov    al, 20h ; 01/08/2020
1519                                     <1>         ;cli     ; disable interrupts till stack cleared
1520                                     <1>         ;out    INTB00, al ; For controll2 #2
1521 00012019 E6A0                      <1>         out     0A0h, al
1522                                     <1>         IRQsrv_6:
1523                                     <1>         ;mov    byte [IRQnum], 0 ; reset
1524                                     <1>         ;;mov    al, EOI ; end of interrupt
1525                                     <1>         ;mov    al, 20h ; 01/08/2020
1526                                     <1>         ;cli     ; disable interrupts till stack cleared
1527                                     <1>         ;out    INTA00, al ; end of interrupt to 8259 - 1
1528 0001201B E620                      <1>         out     20h, al
1529                                     <1>         IRQsrv_7:
1530                                     <1>         ;
1531                                     <1>         ;; 13/06/2017
1532                                     <1>         ;or     word [ebp+8], 200h ; force enabling interrupts
1533 0001201D 8B0D[cc880100]          <1>         ;
1534 00012023 0F22D9                  <1>         mov     ecx, [IRQ_cr3] ; previous content of cr3 register
1535                                     <1>         ; restore cr3 register content
1536 00012026 61                      <1>         popad ; edi,esi,ebp,(crement esp by 4),ebx,edx,ecx,eax
1537                                     <1>         ;
1538 00012027 0FA9                      <1>         pop     gs
1539 00012029 0FA1                      <1>         pop     fs
1540 0001202B 07                      <1>         pop     es
1541 0001202C 1F                      <1>         pop     ds
1542                                     <1>         ;
1543 0001202D CF                      <1>         ;ret     ; return from interrupt
1544                                     <1>
1545                                     <1>         ; 17/04/2021
1546                                     <1>         ; ('get_device_number' procedure is disabled as temporary)
1547                                     <1>
1548                                     <1>         ;get_device_number:
1549                                     <1>         ;
1550                                     <1>         ; 08/10/2016
1551                                     <1>         ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1552                                     <1>         ;
1553                                     <1>         ; This procedure compares name of requested
1554                                     <1>         ; device with kernel device names and
1555                                     <1>         ; installable device names. If names match,
1556                                     <1>         ; the relevant device index (entry) number
1557                                     <1>         ; will be returned the caller (sysopen)
1558                                     <1>         ; for the requested device.
1559                                     <1>         ;
1560                                     <1>         ; NOTE: Installable device drivers must
1561                                     <1>         ; be loaded before using 'sysopen'
1562                                     <1>         ; (opende) system call.
1563                                     <1>         ;
1564                                     <1>         INPUT:
1565                                     <1>         ; ESI = device name address (ASCIIIZ)
1566                                     <1>         ; (in kernel's memory space)
1567                                     <1>         ; max name length = 8 without '/dev/'
1568                                     <1>         ; Device name will be capitalized
1569                                     <1>         ; and if there is, '/dev/' will be
1570                                     <1>         ; removed from name before comparising)
1571                                     <1>         ;
1572                                     <1>         OUTPUT:
1573                                     <1>         ; cf = 0 ->
1574                                     <1>         ; EAX (AL) = device entry/index number
1575                                     <1>         ; cf = 1 -> device not found (installed)
1576                                     <1>         ; or invalid device name
1577                                     <1>         ; (AL=0)
1578                                     <1>         ; device_name = device name address (asciiz)
1579                                     <1>         ;
1580                                     <1>         ; Modified registers: EAX, EBX, ESI, EDI
1581                                     <1>
1582                                     <1>         mov     edi, device_name
1583                                     <1>         call    lods_b_capitalize
1584                                     <1>         mov     ah, al
1585                                     <1>         cmp     al, '/'
1586                                     <1>         jne     short gdn_1
1587                                     <1>         mov     edi, device_name
1588                                     <1>         call    lods_b_capitalize
1589                                     <1>         and     al, al ; 0 ?
1590                                     <1>         jz      short gdn_err ; null name after '/'
1591                                     <1>         gdn_1:
1592                                     <1>         cmp     al, 'D'
1593                                     <1>         jne     short gdn_2
1594                                     <1>         call    lods_b_capitalize
1595                                     <1>         cmp     al, 'E'
1596                                     <1>         jne     short gdn_2
1597                                     <1>         call    lods_b_capitalize
1598                                     <1>         cmp     al, 'V'
1599                                     <1>         jne     short gdn_2
1600                                     <1>         lods_b
1601                                     <1>         cmp     al, '/'
1602                                     <1>         je      short gdn_4
1603                                     <1>         gdn_2:
1604                                     <1>         cmp     ah, '/'
1605                                     <1>         jne     short gdn_5
1606                                     <1>         gdn_err:
1607                                     <1>         ; invalid device name or device not found
1608                                     <1>         xor     eax, eax ; 0
1609                                     <1>         stc
1610                                     <1>         retn
1611                                     <1>         gdn_3:
1612                                     <1>         cmp     al, '/'
1613                                     <1>         jne     short gdn_5
1614                                     <1>         gdn_4:
1615                                     <1>         mov     edi, device_name
1616                                     <1>         jmp     short gdn_6
1617                                     <1>         gdn_5:
1618                                     <1>         cmp     al, 0
1619                                     <1>         je      short gdn_7
1620                                     <1>         gdn_6:
1621                                     <1>         call    lods_b_capitalize
1622                                     <1>         cmp     edi, device_name + 8
1623                                     <1>         jb      short gdn_3
1624                                     <1>         cmp     al, 0
1625                                     <1>         jne     short gdn_err
1626                                     <1>         cmp     edi, device_name + 1
1627                                     <1>         jna     short gdn_err ; null name after '/'
1628                                     <1>         gdn_7:
1629                                     <1>         stosb
1630                                     <1>         ; zero padding ("NAME",0,0,0,0)
1631                                     <1>         cmp     edi, device_name + 8
1632                                     <1>         jb      short gdn_7
1633                                     <1>         gdn_8:
1634                                     <1>         ; search for kernel device names
1635                                     <1>         mov     esi, device_name
1636                                     <1>         mov     edi, KDEV_NAME

```

```

1637      <1> ; xor     eax, eax
1638      <1> ;gdn_9:
1639      <1>      cmpsd
1640      <1>      jne     short gdn_10
1641      <1>      cmpsd
1642      <1>      jne     short gdn_11
1643      <1>      jmp     short gdn_17 ; match
1644      <1> ;gdn_10:
1645      <1>      cmpsd ; add esi, 4 & add edi, 4
1646      <1> ;gdn_11:
1647      <1>      mov     esi, device_name
1648      <1>      inc     al
1649      <1>      cmp     al, NumOfKernelDevNames
1650      <1>      jnb     short gdn_9
1651      <1> ;gdn_12:
1652      <1>      ; search for installable device names
1653      <1>      ; esi = offset device_name
1654      <1>      mov     edi, IDEV_NAME
1655      <1>      sub     al, al ; 0
1656      <1> ;gdn_13:
1657      <1>      cmpsd
1658      <1>      jne     short gdn_14
1659      <1>      cmpsd
1660      <1>      jne     short gdn_15
1661      <1>      jmp     short gdn_19 ; match
1662      <1> ;gdn_14:
1663      <1>      cmpsd ; add esi, 4 & add edi, 4
1664      <1> ;gdn_15:
1665      <1>      mov     esi, device_name
1666      <1>      inc     al
1667      <1>      cmp     al, NumOfInstallableDevices
1668      <1>      jnb     short gdn_13
1669      <1> ;gdn_16:
1670      <1>      ; error: invalid device name (not found) !
1671      <1>      xor     al, al
1672      <1>      stc
1673      <1>      retn
1674      <1> ;gdn_17:
1675      <1>      ; name match (with one of kernel device names)
1676      <1>      ;
1677      <1>      ; convert KDEV_NAME index to
1678      <1>      ; KDEV_NUMBER index
1679      <1>      ; (different names are used for same devices)
1680      <1>      ; (example: "COM1" & "TTY8" = device number 18)
1681      <1>      mov     ebx, eax ; < 256
1682      <1>      mov     al, [KDEV_NUMBER+ebx]
1683      <1>      ;
1684      <1>      ; check if empty dev entry in the list
1685      <1>      cmp     byte [DEV_OPENMODE+eax], 0
1686      <1>      ja      short gdn_18 ; it must be already set
1687      <1>      ;
1688      <1>      ; (re)set device name and access flags
1689      <1>      ; (remain open work will be easy after that)
1690      <1>      ; (NOTE: here, data will be copied to bss section)
1691      <1>      mov     bl, al
1692      <1>      sub     edi, 8 ; kernel device name address (data)
1693      <1>      shl     bx, 2
1694      <1>      mov     [DEV_NAME_PTR+ebx], edi ; (all) device names
1695      <1>      mov     bl, [KDEV_ACCESS+eax] ; kernel dev list (data)
1696      <1>      mov     [DEV_ACCESS+eax], bl ; (all) device list (bss)
1697      <1> ;gdn_18:
1698      <1>      inc     al ; 1 to NumOfKernelDevNames (<=7Fh)
1699      <1>      ; eax = device index/entry number
1700      <1>      retn
1701      <1> ;gdn_19:
1702      <1>      ; name match (with one of installable device names)
1703      <1>      ;
1704      <1>      ; al = 0 to NumOfInstallableDevices - 1 (<=7Fh)
1705      <1>      ;
1706      <1>      mov     ebx, eax
1707      <1>      add     bl, NumOfKernelDevices ; < NUMOFDEVICES
1708      <1>      ;
1709      <1>      ; check if empty dev entry in the list
1710      <1>      cmp     byte [DEV_OPENMODE+ebx], 0
1711      <1>      ja      short gdn_20 ; it must be already set
1712      <1>      ;
1713      <1>      ; (re)set device name and access flags
1714      <1>      ; (remain open work will be easy after that)
1715      <1>      sub     edi, 8 ; installable device name address
1716      <1>      shl     bx, 2 ; *4
1717      <1>      mov     [DEV_NAME_PTR+ebx], edi ; (all) device names
1718      <1>      shr     bx, 2
1719      <1>      mov     al, [IDEV_FLAGS+eax] ; installable dev list
1720      <1>      mov     [DEV_ACCESS+ebx], al ; (all) device list
1721      <1> ;gdn_20:
1722      <1>      mov     al, bl
1723      <1>      ; eax = device index/entry number ; < NUMOFDEVICES
1724      <1>      retn
1725      <1> ;
1726      <1> ;lods_b_capitalize:
1727      <1>      ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1728      <1>      ; INPUT -> [esi] = character
1729      <1>      ;      edi = destination
1730      <1>      ; OUTPUT -> AL contains capitalized character
1731      <1>      ;      esi = esi+1
1732      <1>      ;      edi = edi+1
1733      <1>      ;
1734      <1>      lodsb
1735      <1>      cmp     al, 61h
1736      <1>      jnb     short lods_b_cap_retn
1737      <1>      cmp     al, 7Ah
1738      <1>      ja      short lods_b_cap_retn
1739      <1>      and     al, 0DFh
1740      <1> ;lods_b_cap_retn:
1741      <1>      stosb
1742      <1>      retn
1743      <1> ;
1744      <1> ; 17/04/2021
1745      <1> ; ('device_open' procedure is disabled as temporary)
1746      <1> ;
1747      <1> ;device_open:
1748      <1>      ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)
1749      <1>      ; Complete device opening work for sysopen (device)
1750      <1>      ;
1751      <1>      ; INPUT ->
1752      <1>      ;      EAX = Device Number (AL)
1753      <1>      ;      CL = Open mode (1 = read, 2 = write)
1754      <1>      ;      CH = Device access byte (bit 0 = 0)
1755      <1>      ; OUTPUT ->
1756      <1>      ;      EAX = Device Number
1757      <1>      ;      CF = 0 -> device has been opened
1758      <1>      ;      CF = 1 -> device could not be opened
1759      <1>      ;
1760      <1>      ; Modified registers: ebx, (edx, ecx, esi, edi, ebp)

```



```

1761 <1> ;
1762 <1> ;
1763 <1> mov ebx, eax
1764 <1> shl bx, 2 ; *4
1765 <1> ;
1766 <1> test ch, 80h ; bit 7, installable device driver flag
1767 <1> jz short d_open_2 ; kernel device
1768 <1> ; installable device
1769 <1> d_open_1:
1770 <1> jmp dword [ebx+IDEV_OADDR-4]
1771 <1> d_open_2:
1772 <1> jmp dword [ebx+KDEV_OADDR-4]
1773 <1> ;
1774 <1> ; 17/04/2021
1775 <1> ; ('device_close' procedure is disabled as temporary)
1776 <1> ;
1777 <1> device_close:
1778 <1> ; 08/10/2016 - TRDOS 386 (TRDOS v2.0)
1779 <1> ; Complete device closing work for sysclose (device)
1780 <1> ;
1781 <1> ; INPUT ->
1782 <1> ; EAX = Device Number (AL)
1783 <1> ; CL = Open mode (1 = read, 2 = write)
1784 <1> ; CH = Device access byte (bit 0 = 0)
1785 <1> ; OUTPUT ->
1786 <1> ; EAX = Device Number
1787 <1> ; CF = 0 -> device has been closed
1788 <1> ; CF = 1 -> device could not be closed
1789 <1> ;
1790 <1> ; Modified registers: ebx, (edx, ecx, esi, edi, ebp)
1791 <1> ;
1792 <1> ;
1793 <1> mov ebx, eax
1794 <1> shl bx, 2 ; *4
1795 <1> ;
1796 <1> test ch, 80h ; bit 7, installable device driver flag
1797 <1> jz short d_close_2 ; kernel device
1798 <1> ; installable device
1799 <1> d_close_1:
1800 <1> jmp dword [ebx+IDEV_CADDR-4]
1801 <1> d_close_2:
1802 <1> jmp dword [ebx+KDEV_CADDR-4]
1803 <1> ;
1804 <1> rnull:
1805 <1> ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1806 <1> ; read null (read from null device)
1807 <1> retn
1808 <1> ;
1809 <1> wnull:
1810 <1> ; 07/10/2016 - TRDOS 386 (TRDOS v2.0)
1811 <1> ; write null (write to null device)
1812 <1> retn
1813 <1> ;
1814 <1> dev_IRQ_service:
1815 <1> ; 12/05/2017
1816 <1> ; 13/04/2017
1817 <1> ; 27/02/2017 - TRDOS 386 (TRDOS v2.0)
1818 <1> ; INPUT ->
1819 <1> ; AL = IRQ Number (0 to 15)
1820 <1> ;
1821 <1> push ebx
1822 <1> movzx ebx, al
1823 <1> shl bl, 2 ; * 4
1824 <1> mov ebx, [ebx+DEV_INT_HNDLR]
1825 <1> and ebx, ebx
1826 <1> jz short dIRQ_s_retn
1827 <1> push eax
1828 <1> ;
1829 <1> call ebx
1830 <1> ;
1831 <1> pop eax
1832 <1> dIRQ_s_retn:
1833 <1> pop ebx
1834 <1> retn
1835 <1> ;
1836 <1> set_dev_IRQ_service:
1837 <1> ; 09/08/2022
1838 <1> ; 29/07/2022 (TRDOS 386 Kernel v2.0.5)
1839 <1> ; 13/04/2017 - TRDOS 386 (TRDOS v2.0)
1840 <1> ;
1841 <1> ; Set Device Interrupt Service
1842 <1> ;
1843 <1> ; INPUT ->
1844 <1> ; AL = IRQ Number
1845 <1> ; EBX = Hardware Interrupt Service Address
1846 <1> ;
1847 <1> ; Note: There is not a validation check here
1848 <1> ; because this procedure is called by
1849 <1> ; TRDOS 386 kernel !
1850 <1> ; (Even if a device driver does not exist
1851 <1> ; this setting may be used by sysaudio
1852 <1> ; and other system calls for hardware
1853 <1> ; components which use IRQ method for I/O.)
1854 <1> ;
1855 <1> ;push esi
1856 <1> movzx esi, al
1857 <1> shl si, 2 ; * 4
1858 <1> ; 09/08/2022
1859 <1> shl esi, 2 ; * 4
1860 <1> mov [esi+DEV_INT_HNDLR], ebx
1861 <1> pop esi
1862 <1> retn
1863 <1> ;
1864 <1> sysaudio: ; AUDIO FUNCTIONS
1865 <1> ; 28/01/2025
1866 <1> ; 27/01/2025
1867 <1> ; 15/01/2025
1868 <1> ; 12/01/2025
1869 <1> ; 11/01/2025 (TRDOS 386 v2.0.10)
1870 <1> ; 29/12/2024
1871 <1> ; 19/12/2024
1872 <1> ; 23/08/2024 (TRDOS 386 v2.0.9)
1873 <1> ; 05/06/2024
1874 <1> ; 04/06/2024
1875 <1> ; 23/05/2024 (TRDOS 386 v2.0.8)
1876 <1> ; 19/11/2023 (TRDOS 386 v2.0.7)
1877 <1> ; 29/07/2022 (TRDOS 386 v2.0.5)
1878 <1> ; 12/02/2021 (TRDOS 386 v2.0.3)
1879 <1> ; 28/07/2020
1880 <1> ; 27/07/2020
1881 <1> ; 10/10/2017
1882 <1> ; 22/06/2017
1883 <1> ; 28/05/2017, 04/06/2017, 05/06/2017, 10/06/2017
1884 <1> ; 01/05/2017, 12/05/2017, 15/05/2017, 20/05/2017

```

```

1885 <1> ; 21/04/2017, 22/04/2017, 23/04/2017, 24/04/2017
1886 <1> ; 10/04/2017, 13/04/2017, 14/04/2017, 16/04/2017
1887 <1> ; 03/04/2017 (VIA VT8237R)
1888 <1> ; 01/04/2016 (trdosk6.s -> tdosk8.s)
1889 <1> ; 16/05/2016 - TRDOS 386 (TRDOS v2.0)
1890 <1>
1891 <1> Inputs:
1892 <1>
1893 <1> BH = 0 -> Beep (PC Speaker)
1894 <1> BL = Duration Counter (1 for 1/64 second)
1895 <1> CX = Frequency Divisor (1193180/Frequency)
1896 <1> (1331 for 886 Hz)
1897 <1>
1898 <1> 01/04/2017
1899 <1>
1900 <1> BH = 1 -> DETECT (& ENABLE) AUDIO DEVICE
1901 <1> BL = 0 : PC SPEAKER
1902 <1> 1 : SOUND BLASTER 16
1903 <1> 2 : INTEL AC'97
1904 <1> 3 : VIA VT8237R (VT8233)
1905 <1> 4 : INTEL HDA
1906 <1> 5-FEH : unknown/invalid
1907 <1> ; 04/06/2017
1908 <1> FFh : Get current audio device id
1909 <1>
1910 <1> BH = 2 -> ALLOCATE AUDIO BUFFER (for user)
1911 <1> ECX = Audio Buffer Size (must be equal to
1912 <1> the half of DMA buffer size)
1913 <1> EDX = Virtual Address of the buffer
1914 <1> (This is not DMA buffer!)
1915 <1>
1916 <1> BH = 3 -> INITIALIZE AUDIO DEVICE
1917 <1> BL = 0,2 -> for Signal Response Byte
1918 <1> CL = Signal Response Byte Value (fixed)
1919 <1> if BL = 0
1920 <1> auto increment of S.R.B. value
1921 <1> if BL = 2
1922 <1> EDX = Signal Response (Return) Byte Address
1923 <1>
1924 <1> BL = 1 for CallBack Method
1925 <1> EDX = CallBack Service Address (virtual)
1926 <1>
1927 <1> BL > 2 -> invalid function
1928 <1>
1929 <1> (Audio buffer must be allocated before
1930 <1> initialization.)
1931 <1>
1932 <1> BH = 4 -> START TO PLAY
1933 <1> BL = Mode
1934 <1> Bit 0 = mono/stereo (1 = stereo)
1935 <1> Bit 1 = 8 bit / 16 bit (1 = 16 bit)
1936 <1> CX = Sampling Rate (Hz)
1937 <1>
1938 <1> BH = 5 -> PAUSE
1939 <1> BL = Any
1940 <1>
1941 <1> BH = 6 -> CONTINUE TO PLAY
1942 <1> BL = Any
1943 <1>
1944 <1> BH = 7 -> STOP
1945 <1> BL = Any
1946 <1>
1947 <1> BH = 8 -> RESET
1948 <1> BL = Any
1949 <1>
1950 <1> BH = 9 -> CANCEL (CALLBACK or S.R.B. SERVICE)
1951 <1> BL = Any
1952 <1>
1953 <1> BH = 10 -> DEALLOCATE AUDIO BUFFER (for user)
1954 <1> BL = Any
1955 <1>
1956 <1> BH = 11 -> SET VOLUME LEVEL
1957 <1> BL: (Bit 0 to 6)
1958 <1> 0 = Master (Playback, Lineout) volume
1959 <1> 1 = PCM out volume ; 23/05/2024
1960 <1> CL = Left Channel volume (0 to 31 max)
1961 <1> CH = Right Channel Volume (0 to 31 max)
1962 <1>
1963 <1> Note: If BL >= 80h (Bit 7 of BL is set),
1964 <1> volume level will be set for next playing
1965 <1> (actual volume level will not be changed
1966 <1> immediately)
1967 <1>
1968 <1> BH = 12 -> DISABLE AUDIO DEVICE
1969 <1> (reset audio device and unlink dma buffer)
1970 <1> BL = Any
1971 <1>
1972 <1> 12/05/2017
1973 <1> BH = 13 -> MAP DMA BUFFER TO USER
1974 <1> (for direct access to system's dma buffer)
1975 <1>
1976 <1> ECX = map size in bytes
1977 <1> (will be rounded up to page borders)
1978 <1> EDX = Virtual Address of the buffer
1979 <1> (will be rounded up to page borders)
1980 <1>
1981 <1> 05/06/2017
1982 <1> 04/06/2017
1983 <1> BH = 14 -> GET AUDIO DEVICE INFO
1984 <1> BL: 0 = Audio Controller Info
1985 <1> ; 19/11/2023
1986 <1> BL: 1 = Audio (AC'97) Codec Info
1987 <1> BL > 1 = Invalid for now!
1988 <1>
1989 <1> 22/06/2017
1990 <1> BH = 15 -> GET CURRENT SOUND DATA (for graphics)
1991 <1> BL: 0 -> PCM OUT data
1992 <1> > 0 -> Invalid for now!
1993 <1> ECX = 0 -> Get DMA Buffer Pointer
1994 <1> EDX = Not Used
1995 <1> ECX > 0 -> Byte count for buffer (EDX)
1996 <1> EDX = Buffer Address (virtual)
1997 <1>
1998 <1> 10/10/2017
1999 <1> BH = 16 -> UPDATE DMA BUFFER DATA
2000 <1> (by using the Audio Buffer content)
2001 <1> BL = 0 : Update dma half buffer in sequence
2002 <1> (automatic destination)
2003 <1> 1 : Update 1st half of the dma buffer
2004 <1> 2 : Update 2nd half of the dma buffer
2005 <1> 3-FEH: Invalid!
2006 <1> FFh = Get current flag value
2007 <1> (Half buffer number -1)
2008 <1>

```

```

2009      <1>      ;      24/05/2024
2010      <1>      ;      BH = 17 -> GET VOLUME LEVEL
2011      <1>      ;      BL: 0 = Master (Playback, Lineout) volume
2012      <1>      ;      1 = PCM out volume
2013      <1>      ;
2014      <1>      ;      Outputs:
2015      <1>      ;
2016      <1>      ;      For BH = 0 -> Beep
2017      <1>      ;      None
2018      <1>      ;
2019      <1>      ;      01/04/2017
2020      <1>      ;
2021      <1>      ;      For BH = 1 -> DETECT (& ENABLE) AUDIO DEVICE
2022      <1>      ;      AH = 0 : PC SPEAKER
2023      <1>      ;      1 : SOUND BLASTER 16
2024      <1>      ;      2 : INTEL AC'97
2025      <1>      ;      3 : VIA VT8237R (VT8233)
2026      <1>      ;      4 : INTEL HDA
2027      <1>      ;      5-FFh : unknown/invalid
2028      <1>      ;      AL = mode status
2029      <1>      ;      bit 0 = mono /stereo (1 = stereo)
2030      <1>      ;      bit 1 = 8 bit / 16 bit (1 = 16 bit)
2031      <1>      ;      04/06/2017
2032      <1>      ;      EBX = PCI DEVICE/VENDOR ID (if >0)
2033      <1>      ;      (BX = VENDOR ID)
2034      <1>      ;      (if CF = 1 -> Error code in EAX)
2035      <1>      ;
2036      <1>      ;      For BH = 2 -> ALLOCATE AUDIO BUFFER (for user)
2037      <1>      ;      EAX = Physical Address of the buffer
2038      <1>      ;      (if CF = 1 -> Error code in EAX)
2039      <1>      ;
2040      <1>      ;      For BH = 3 -> INITIALIZE AUDIO DEVICE
2041      <1>      ;      (if CF = 1 -> Error code in EAX)
2042      <1>      ;
2043      <1>      ;      For BH = 4 -> START TO PLAY
2044      <1>      ;      none (if CF = 1 -> Error code in EAX)
2045      <1>      ;
2046      <1>      ;      For BH = 5 -> PAUSE
2047      <1>      ;      none (if CF = 1 -> Error code in EAX)
2048      <1>      ;
2049      <1>      ;      For BH = 6 -> CONTINUE TO PLAY
2050      <1>      ;      none (if CF = 1 -> Error code in EAX)
2051      <1>      ;
2052      <1>      ;      For BH = 7 -> STOP
2053      <1>      ;      none (if CF = 1 -> Error code in EAX)
2054      <1>      ;
2055      <1>      ;      For BH = 8 -> RESET
2056      <1>      ;      none (if CF = 1 -> Error code in EAX)
2057      <1>      ;
2058      <1>      ;      For BH = 9 -> CANCEL (CALLBACK or S.R.B. SERVICE)
2059      <1>      ;      none (if CF = 1 -> Error code in EAX)
2060      <1>      ;
2061      <1>      ;      For BH = 10 -> DEALLOCATE AUDIO BUFFER (for user)
2062      <1>      ;      none (if CF = 1 -> Error code in EAX)
2063      <1>      ;
2064      <1>      ;      For BH = 11 -> SET VOLUME LEVEL
2065      <1>      ;      none (if CF = 1 -> Error code in EAX)
2066      <1>      ;
2067      <1>      ;      For BH = 12 -> DISABLE AUDIO DEVICE
2068      <1>      ;      none (if CF = 1 -> Error code in EAX)
2069      <1>      ;
2070      <1>      ;      12/05/2017
2071      <1>      ;      For BH = 13 -> MAP DMA BUFFER TO USER
2072      <1>      ;      EAX = Physical Address of the buffer
2073      <1>      ;      (if CF = 1 -> Error code in EAX)
2074      <1>      ;
2075      <1>      ;      04/06/2017
2076      <1>      ;      For BH = 14 -> GET AUDIO DEVICE INFO
2077      <1>      ;      (for BL = 0) ; 05/06/2017
2078      <1>      ;      EAX = IRQ Number in AL
2079      <1>      ;      Audio Device Number in AH
2080      <1>      ;      EBX = DEV/VENDOR ID
2081      <1>      ;      (DDDDDDDDDDDDDDDDVVVVVVVVVVVVVVVV)
2082      <1>      ;      ECX = BUS/DEV/FN
2083      <1>      ;      (00000000BBBBBBBDDDDDDFF00000000)
2084      <1>      ;      EDX = NAMBAR/NAMBAR (for AC97)
2085      <1>      ;      (Low word, DX = NAMBAR address)
2086      <1>      ;      EDX = Base IO Addr (DX) for SB16 & VT8233
2087      <1>      ;      (if CF = 1 -> Error code in EAX)
2088      <1>      ;      (ERR_DEV_NOT_RDY = 15)
2089      <1>      ;      (for BL = 1) -for AC97- ; 19/11/2023
2090      <1>      ;      EAX = Extended Audio ID (MX28) in AX
2091      <1>      ;      AX bit 0 - VRA bit
2092      <1>      ;      HW of EAX = PCM output sample rate (HZ)
2093      <1>      ;      EBX = VENDOR ID 1 (BX), VENDOR ID 2 (HW)
2094      <1>      ;      (if CF = 1 -> Error code in EAX)
2095      <1>      ;      (ERR_DEV_NOT_RDY = 15)
2096      <1>      ;      Note: EAX & EBX = 0 (For SB16,VIA,HDA)
2097      <1>      ;
2098      <1>      ;      22/06/2017
2099      <1>      ;      For BH = 15 -> GET CURRENT SOUND DATA
2100      <1>      ;      (for graphics)
2101      <1>      ;      (for BL = 0)
2102      <1>      ;      If ECX input is 0
2103      <1>      ;      EAX = DMA Buffer Current Position (Offset)
2104      <1>      ;      If ECX input > 0
2105      <1>      ;      EAX = Actual transfer count
2106      <1>      ;      (Sound samples will be copied from
2107      <1>      ;      Current DMA Buffer Position to EDX
2108      <1>      ;      virtual address as EAX bytes.)
2109      <1>      ;      ((If CF = 1 -> Error code in EAX))
2110      <1>      ;
2111      <1>      ;      10/10/2017
2112      <1>      ;      For BH = 16 -> UPDATE DMA BUFFER DATA
2113      <1>      ;      EAX = 0, if the updated (or current)
2114      <1>      ;      half buffer is DMA half buffer 1
2115      <1>      ;      EAX = 1, if the updated (or current)
2116      <1>      ;      half buffer is DMA half buffer 2
2117      <1>      ;      (If CF = 1 -> Error code in EAX)
2118      <1>      ;
2119      <1>      ;      24/05/2024
2120      <1>      ;      For BH = 17 -> GET VOLUME LEVEL
2121      <1>      ;      IF BL input is 0
2122      <1>      ;      Master (Playback, Lineout) volume
2123      <1>      ;      If BL input is 1
2124      <1>      ;      PCM out volume
2125      <1>      ;      If BL input > 1
2126      <1>      ;      Invalid for now ; 24/05/2024
2127      <1>      ;
2128      <1>      ;      CL = Left Channel volume (0 to 31 max)
2129      <1>      ;      CH = Right Channel volume (0 to 31 max)
2130      <1>      ;
2131      <1>      ;
2132      00012052 80FF12      <1>      cmp      bh, AUDIO1L/4

```

```

2133 00012055 0F8323AAFFFF <1> jnb sysret
2134 <1>
2135 0001205B C0E702 <1> shl bh, 2 ; *4
2136 0001205E 0FB6F7 <1> movzx esi, bh
2137 <1>
2138 <1> ; 22/04/2017
2139 00012061 31C0 <1> xor eax, eax
2140 00012063 A3[548E0100] <1> mov [u.r0], eax ; 0
2141 <1>
2142 00012068 FF96[73200100] <1> call dword [esi+AUDIO1]
2143 <1> ;jc error
2144 0001206E E90BAAFFFF <1> jmp sysret
2145 <1>
2146 00012073 [BD230000] <1> AUDIO1: dd _beep ; 12/02/2021
2147 <1> ;dd beep ; FUNCTION = 0 (b1 = Duration Counter
2148 <1> ; cx = Frequency Divisor
2149 00012077 [BB200100] <1> dd soundc_detect
2150 0001207B [50210100] <1> dd sound_alloc
2151 0001207F [13220100] <1> dd soundc_init
2152 00012083 [F7230100] <1> dd sound_play
2153 00012087 [84240100] <1> dd sound_pause
2154 0001208B [AE240100] <1> dd sound_continue
2155 0001208F [D8240100] <1> dd sound_stop
2156 00012093 [03250100] <1> dd soundc_reset
2157 00012097 [36250100] <1> dd soundc_cancel
2158 0001209B [5C250100] <1> dd sound_dalloc
2159 0001209F [87250100] <1> dd sound_volume
2160 000120A3 [C7250100] <1> dd soundc_disable
2161 000120A7 [45260100] <1> dd sound_dma_map
2162 000120AB [B3260100] <1> dd soundc_info
2163 000120AF [37270100] <1> dd sound_data
2164 000120B3 [D9270100] <1> dd sound_update
2165 000120B7 [5B280100] <1> dd sound_getvol ; 24/05/2024
2166 <1> %if 0
2167 <1> dd sound_dmaclear ; 29/12/2024
2168 <1> %endif
2169 <1>
2170 <1> AUDIO1LL EQU $ - AUDIO1
2171 <1>
2172 <1> soundc_detect:
2173 <1> ; FUNCTION = 1
2174 <1> ; b1 = Audio device type number
2175 <1> ; (0 = pc speaker, 1 = sound blaster 16, 2 = intel ac97
2176 <1> ; 3 = via vt823x, 4 = intel HDA, 0FFh = any)
2177 <1>
2178 <1> ; 04/06/2017
2179 <1> ;mov ah, [audio_device]
2180 <1> ; 11/01/2025
2181 000120BB A0[D1880100] <1> mov al, [audio_device]
2182 000120C0 80FBFF <1> cmp b1, 0FFh ; get current audio device id
2183 000120C3 7408 <1> je short sysaudio0
2184 <1>
2185 <1> ;and ah, ah
2186 <1> ; 11/01/2025
2187 000120C5 20C0 <1> and al, al
2188 000120C7 741F <1> jz short soundc_get_dev
2189 <1>
2190 <1> ;cmp ah, b1
2191 <1> ; 11/01/2025
2192 000120C9 38D8 <1> cmp al, b1
2193 000120CB 7568 <1> jne short soundc_dev_err
2194 <1>
2195 <1> sysaudio0:
2196 <1> ;mov al, [audio_mode]
2197 <1> ; 11/01/2025
2198 000120CD 8A25[09890100] <1> mov ah, [audio_mode]
2199 <1> sysaudio1:
2200 000120D3 A3[548E0100] <1> mov [u.r0], eax
2201 000120D8 8B1D[DC880100] <1> mov ebx, [audio_vendor] ; (DEVICE/VENDOR ID)
2202 000120DE 8B2D[508E0100] <1> mov ebp, [u.usp]
2203 000120E4 895D10 <1> mov [ebp+16], ebx ; ebx
2204 000120E7 C3 <1> retn
2205 <1>
2206 <1> soundc_get_dev:
2207 <1> ; 28/05/2017
2208 <1> ; 03/04/2017, 24/04/2017
2209 000120E8 C605[D0880100]00 <1> mov byte [audio_pci], 0
2210 000120EF 80FB03 <1> cmp b1, 3 ; VIA VT8233 (VT8237R) Audio Controller & AC97 Codec
2211 <1> ;jne short soundc_get_dev_sb
2212 <1> ; 28/05/2017
2213 000120F2 7220 <1> jb short soundc_get_dev_sb
2214 000120F4 773F <1> ja short soundc_dev_err ; temporary (28/05/2017)
2215 <1>
2216 000120F6 E892180000 <1> call DetectVT8233
2217 000120FB 7238 <1> jc short soundc_dev_err
2218 <1> ; eax = 0
2219 <1>
2220 <1> ;mov ebx, [audio_vendor]
2221 <1> ; ebx = DEVICE/VENDOR ID
2222 <1> ;
2223 <1> ; DDDDDDDDDDDDDDDVVVVVVVVVVVVVVVVVV
2224 000120FD B003 <1> mov al, 3 ; VIA VT8237R (VT3233) Audio Controller
2225 000120FF 88C4 <1> mov ah, al
2226 <1>
2227 <1> soundc_get_pci_dev_ok: ; 28/05/2017
2228 00012101 FE05[D0880100] <1> inc byte [audio_pci] ; = 1
2229 <1> soundc_get_dev_ok:
2230 <1>
2231 <1> soundc_get_dev_sb16_ok:
2232 00012107 A2[D1880100] <1> mov [audio_device], al
2233 0001210C 8825[09890100] <1> mov [audio_mode], ah ; stereo (bit0), 16 bit (bit1) capability
2234 00012112 EBBF <1> jmp short sysaudio1
2235 <1>
2236 <1> soundc_get_dev_sb:
2237 <1> ; 24/04/2017
2238 00012114 80FB01 <1> cmp b1, 1 ; Sound Blaster 16
2239 00012117 750E <1> jne short soundc_get_dev_ich ; 28/05/2017
2240 <1>
2241 00012119 E8431D0000 <1> call DetectSB
2242 0001211E 7215 <1> jc short soundc_dev_err
2243 00012120 B801030000 <1> mov eax, 0301h ; Sound Blaster 16
2244 00012125 EBEO <1> jmp short soundc_get_dev_sb16_ok
2245 <1>
2246 <1> soundc_get_dev_ich:
2247 <1> ; 28/05/2017
2248 <1> ;cmp b1, 2 ; Intel AC'97 Audio Controller (ICH)
2249 <1> ;jne short soundc_dev_err ; Temporary (28/05/2017)
2250 <1> ;
2251 <1> ; (Here will be modified just after
2252 <1> ; new sound card code will be ready!)
2252 00012127 E84C180000 <1> call DetectICH
2253 0001212C 7207 <1> jc short soundc_dev_err
2254 <1>
2255 0001212E B802030000 <1> mov eax, 0302h ; AC'97 (ICH)
2256 00012133 EBCC <1> jmp short soundc_get_pci_dev_ok

```

```

2257 <1>
2258 <1> soundc_dev_err:
2259 00012135 B80F000000 <1> mov     eax, ERR_DEV_NOT_RDY ; Device not ready !
2260 <1> ; 29/07/2022
2261 <1> ;sub     eax, eax
2262 <1> ;mov     al, ERR_DEV_NOT_RDY
2263 0001213A EB05 <1> jmp      short sysaudio_err
2264 <1>
2265 <1> soundc_respond_err:
2266 <1> ; ERR_TIME_OUT ; 'time out !' error
2267 0001213C B819000000 <1> mov     eax, ERR_DEV_NOT_RESP ; 'device not responding !' error
2268 <1> ; 29/07/2022
2269 <1> ;sub     eax, eax
2270 <1> ;mov     al, ERR_DEV_NOT_RESP
2271 <1> sysaudio_err:
2272 00012141 A3[548E0100] <1> mov     [u.r0], eax
2273 00012146 A3[C08E0100] <1> mov     [u.error], eax
2274 0001214B E90EA9FFFF <1> jmp      error
2275 <1>
2276 <1> sound_alloc:
2277 <1> ; FUNCTION = 2
2278 <1> ; ecx = audio buffer size (in bytes)
2279 <1> ; edx = audio buffer address (virtual)
2280 <1> ; 28/01/2025
2281 <1> ; 27/01/2025 (BugFix)
2282 <1> ; 15/01/2025
2283 <1> ; 12/01/2025
2284 <1> ; 25/11/2023
2285 <1> ; 27/07/2020
2286 <1> ; 28/05/2027
2287 <1> ; 01/05/2017, 15/05/2017
2288 <1> ; 21/04/2017, 24/04/2017
2289 00012150 803D[D0880100]00 <1> cmp     byte [audio_pci], 0
2290 00012157 770F <1> ja      short snd_alloc_0
2291 <1> ; Max. 64KB DMA buffer !!!
2292 00012159 81F900800000 <1> cmp     ecx, 32768
2293 0001215F 760F <1> jna     short snd_alloc_6
2294 <1> ; 25/11/2023
2295 <1> sound_buff_error:
2296 00012161 B82E000000 <1> mov     eax, ERR_BUFFER ; Buffer error !
2297 <1> ; 29/07/2022
2298 <1> ;sub     eax, eax
2299 <1> ;mov     al, ERR_BUFFER
2300 00012166 EBD9 <1> jmp      short sysaudio_err
2301 <1> snd_alloc_0:
2302 <1> ; 25/11/2023
2303 <1> ; Max 128KB DMA buffer size (2 half buffers)
2304 00012168 81F900000100 <1> cmp     ecx, 65536
2305 0001216E 77F1 <1> ja      short sound_buff_error
2306 <1> snd_alloc_6:
2307 <1> ; 15/05/2017
2308 <1> ;cmp     ecx, 4096 ; PAGE_SIZE
2309 <1> ;jb      short sound_buff_error
2310 <1> ; 15/01/2025
2311 00012170 81F940010000 <1> cmp     ecx, 320
2312 00012176 72E9 <1> jnb     short sound_buff_error
2313 <1> ;
2314 <1> ; 28/01/2025
2315 <1> ; 12/01/2025
2316 <1> ;mov     ebx, edx ; (new) virtual address of audio_buffer
2317 <1> ;
2318 00012178 A1[E4880100] <1> mov     eax, [audio_buffer] ; audio buffer address (current)
2319 0001217D 09C0 <1> or      eax, eax
2320 <1> ;jz      short snd_alloc_2
2321 <1> ; 12/01/2025
2322 0001217F 743E <1> jz      short snd_alloc_1 ; first time
2323 <1> ; audio buffer exists !
2324 00012181 8A1D[A58E0100] <1> mov     bl, [u.uno]
2325 00012187 3A1D[FD880100] <1> cmp     bl, [audio_user]
2326 <1> ;jne     sndc_owner_error ; not owner !
2327 <1> ; 25/11/2023
2328 <1> ;je      short snd_alloc_7
2329 <1> ; 12/01/2025
2330 0001218D 7405 <1> je      short snd_alloc_2
2331 0001218F E9F3000000 <1> jmp     sndc_owner_error ; not owner !
2332 <1>
2333 <1> ; 12/01/2025 - TRDOS 386 v2.0.10
2334 <1> ; Important modification:
2335 <1> ;; Always deallocate user's audio_buffer address with
2336 <1> ;; buffer size and then allocate a new one
2337 <1> ;; (address range may be used before)
2338 <1> snd_alloc_2:
2339 <1> ;mov     ecx, [audio_buff_size]
2340 <1> ; 28/01/2025
2341 00012194 870D[EC880100] <1> xchg    ecx, [audio_buff_size] ; *
2342 0001219A 89C3 <1> mov     ebx, eax ; audio buffer address (current)
2343 <1> ;snd_alloc_7:
2344 <1> ; 12/01/2025
2345 <1> ;cmp     eax, edx ; same virtual buffer address ?
2346 <1> ;jne     short snd_alloc_1
2347 <1> ;cmp     ecx, [audio_buff_size]
2348 <1> ;je      short snd_alloc_3 ; Nothing to do !
2349 <1> ;
2350 <1> ; Buffer has been set already!
2351 <1> ;snd_alloc_1:
2352 <1> ; 28/01/2025
2353 0001219C 52 <1> push    ecx
2354 <1> push    edx
2355 <1> ; 12/01/2025
2356 <1> ;mov     ebx, eax ; audio buffer address (current)
2357 <1> ;mov     ecx, [audio_buff_size]
2358 <1> ; ebx = audio buffer address (virtual)
2359 <1> ; ecx = audio buffer size in bytes
2359 0001219D E8EF3EFFFF <1> call    deallocate_user_pages
2360 000121A2 5A <1> pop     edx
2361 <1> pop     ecx
2362 <1> ; 28/01/2025
2363 000121A3 8B0D[EC880100] <1> mov     ecx, [audio_buff_size] ; * ; requested size
2364 000121A9 31C0 <1> xor     eax, eax ; 0
2365 000121AB A3[E4880100] <1> mov     [audio_buffer], eax ; 0
2366 000121B0 A3[E8880100] <1> mov     [audio_p_buffer], eax ; 0
2367 000121B5 A3[EC880100] <1> mov     [audio_buff_size], eax
2368 000121BA A2[FD880100] <1> mov     [audio_user], al ; 0
2369 <1>
2370 <1> ; 27/01/2025 (BugFix)
2371 <1> snd_alloc_1:
2372 <1> ; 12/01/2025
2373 <1> ;snd_alloc_2:
2374 000121BF 89D3 <1> mov     ebx, edx
2375 <1> ; 01/05/2017
2376 <1> ;mov     edx, ~PAGE_OFF ; truncating page offsets
2377 <1> ;
2378 <1> ;and     eax, edx
2379 <1> ;and     ebx, edx
2380 <1> ; 26/11/2023

```

```

2381                                     <1> ;and ecx, edx
2382 000121C1 81E300F0FFFF <1> and ebx, ~PAGE_OFF
2383 000121C7 89CA <1> mov edx, ecx
2384 <1> ; 15/05/2017
2385 <1> ; EAX = Beginning address (physical)
2386 <1> ; EAX = 0 -> Allocate mem block from the 1st proper aperture
2387 <1> ; ECX = Number of bytes to be allocated
2388 000121C9 E86B3BFFFF <1> call allocate_memory_block
2389 000121CE 7291 <1> jc short sound_buff_error
2390 <1>
2391 <1> ; EAX = Physical address of the allocated memory block
2392 <1> ; ECX = Allocated bytes (as rounded up to page border)
2393 <1> ; EBX = Virtual address (as truncated to page border)
2394 000121D0 52 <1> push edx ; 26/11/2023
2395 000121D1 50 <1> push eax
2396 000121D2 53 <1> push ebx
2397 000121D3 51 <1> push ecx
2398 000121D4 E8A03FFFFF <1> call allocate_user_pages
2399 000121D9 59 <1> pop ecx
2400 000121DA 5B <1> pop ebx
2401 000121DB 58 <1> pop eax
2402 000121DC 5A <1> pop edx ; 26/11/2023
2403 000121DD 722A <1> jc short snd_alloc_4 ; insufficient memory, buff error
2404 <1> ; eax = physical address of the user's audio buffer
2405 <1> ; ebx = virtual address of the user's audio buffer
2406 <1> ; 26/11/2023
2407 <1> ; ecx = allocated buffer size (in bytes)
2408 <1> ; -rounded up to page boundary-
2409 <1> ; edx = requested buffer size (in bytes)
2410 000121DF A3[E8880100] <1> mov [audio_p_buffer], eax
2411 000121E4 891D[E4880100] <1> mov [audio_buffer], ebx
2412 <1> ; 26/11/2023
2413 <1> ; mov ecx, edx
2414 <1> ; 25/11/2023
2415 <1> ; cmp ecx, 65536
2416 <1> ; jb short snd_alloc_5
2417 <1> ; dec ecx
2418 <1> ; ecx = 65535
2419 <1> ; (DMA half buffer's sample count must be < 65536)
2420 <1> ; snd_alloc_5:
2421 <1> ; mov [audio_buff_size], ecx
2422 <1> ; 26/11/2023
2423 000121EA 8915[EC880100] <1> mov [audio_buff_size], edx
2424 <1> ;
2425 000121F0 8A15[A58E0100] <1> mov dl, [u.uno]
2426 000121F6 8815[FD880100] <1> mov [audio_user], dl
2427 000121FC A3[548E0100] <1> mov [u.r0], eax
2428 <1> snd_alloc_3:
2429 <1> ; 27/07/2020
2430 00012201 C605[FC880100]00 <1> mov byte [audio_flag], 0 ; clear dma half buffer flag
2431 <1> ;
2432 00012208 C3 <1> retn
2433 <1> snd_alloc_4:
2434 <1> ; 15/05/2017
2435 <1> ; EAX = Beginning address (physical)
2436 <1> ; ECX = Number of bytes to be deallocated
2437 00012209 E8453DFFFF <1> call deallocate_memory_block
2438 0001220E E94EFFFFFF <1> jmp sound_buff_error ; insufficient memory, buff error
2439 <1>
2440 <1> soundc_init:
2441 <1> ; FUNCTION = 3
2442 <1> ; b1 = method (0= s.r.b., 1= callback, 2= auto incr s.r.b.)
2443 <1> ; c1 = signal response byte (initial or fixed) value
2444 <1> ; edx = signal response byte or callback address
2445 <1>
2446 <1> ; 11/01/2025
2447 <1> ; 05/06/2024
2448 <1> ; 04/06/2024
2449 <1> ; 02/12/2023, 04/12/2022
2450 <1> ; 29/07/2022, 07/08/2022
2451 <1> ; 27/07/2020
2452 <1> ; 12/05/2017, 20/05/2017, 28/05/2027
2453 <1> ; 22/04/2017, 23/04/2017, 24/04/2017
2454 <1> ; 13/04/2017, 14/04/2017, 16/04/2017, 21/04/2017
2455 <1> ; 03/04/2017, 10/04/2017
2456 <1>
2457 00012213 A0[D1880100] <1> mov al, [audio_device]
2458 00012218 20C0 <1> and al, al
2459 0001221A 754A <1> jnz short sndc_init6
2460 <1> ;
2461 0001221C C605[D0880100]00 <1> mov byte [audio_pci], 0
2462 00012223 52 <1> push edx
2463 00012224 53 <1> push ebx
2464 00012225 51 <1> push ecx
2465 00012226 E8361C0000 <1> call DetectsB
2466 0001222B 7213 <1> jc short sndc_init8
2467 0001222D 66B80103 <1> mov ax, 0301h ; Sound Blaster 16
2468 00012231 EB1E <1> jmp short sndc_init7
2469 <1>
2470 <1> sndc_init11:
2471 <1> ; 28/05/2017
2472 00012233 E840170000 <1> call DetectICH ; Detect AC'97 (ICH) Audio Controller
2473 00012238 7217 <1> jc short sndc_init7
2474 0001223A 66B80203 <1> mov ax, 0302h ; Intel AC'97 Audio Device
2475 0001223E EB0B <1> jmp short sndc_init12 ; (PCI device)
2476 <1>
2477 <1> sndc_init8:
2478 00012240 E848170000 <1> call DetectVT8233
2479 <1> ; jc short sndc_init7
2480 <1> ; 29/07/2022
2481 00012245 72EC <1> jc short sndc_init11 ; 28/05/2017
2482 <1>
2483 <1> ; eax = 0
2484 00012247 B003 <1> mov al, 3 ; VIA VT8237R (VT8233) Audio Controller
2485 00012249 88C4 <1> mov ah, al
2486 <1>
2487 <1> sndc_init12:
2488 0001224B FE05[D0880100] <1> inc byte [audio_pci] ; = 1
2489 <1> sndc_init7:
2490 00012251 59 <1> pop ecx
2491 00012252 5B <1> pop ebx
2492 00012253 5A <1> pop edx
2493 <1> ; jc soundc_dev_err
2494 <1> ; 29/07/2022
2495 00012254 7305 <1> jnc short sndc_init14
2496 00012256 E9DAFEFFFF <1> jmp soundc_dev_err
2497 <1> sndc_init14:
2498 0001225B A2[D1880100] <1> mov [audio_device], al
2499 00012260 8825[09890100] <1> mov [audio_mode], ah ; stereo (bit0), 16 bit (bit1) capability
2500 <1>
2501 <1> sndc_init6:
2502 00012266 833D[E4880100]00 <1> cmp dword [audio_buffer], 0
2503 <1> ; jna sound_buff_error
2504 <1> ; 29/07/2022

```

```

2505 0001226D 7705      <1>      ja      short sndc_init19
2506 0001226F E9EDFEFFFF <1>      jmp     sound_buff_error ; 07/08/2022
2507                                <1> sndc_init19:
2508 00012274 A0[A58E0100] <1>      mov     al, [u.uno]
2509 00012279 8A25[FD880100] <1>      mov     ah, [audio_user]
2510 0001227F 08E4      <1>      or      ah, ah
2511 00012281 7418      <1>      jz      short sndc_init0
2512 00012283 38E0      <1>      cmp     al, ah
2513 00012285 7419      <1>      je      short sndc_init1
2514                                <1>
2515                                <1> sndc_owner_error:
2516 00012287 B80B000000 <1>      mov     eax, ERR_NOT_OWNER ; 'permission denied !' error
2517                                <1>      ; 29/07/2022
2518                                <1>      ;sub     eax, eax
2519                                <1>      ;mov     al, ERR_NOT_OWNER
2520                                <1> sndc_perm_error:
2521 0001228C A3[548E0100] <1>      mov     [u.r0], eax
2522 00012291 A3[C08E0100] <1>      mov     [u.error], eax
2523 00012296 E9C3A7FFFF <1>      jmp     error
2524                                <1> sndc_init0:
2525 0001229B A2[FD880100] <1>      mov     [audio_user], al
2526                                <1> sndc_init1:
2527 000122A0 8915[00890100] <1>      mov     [audio_cb_addr], edx
2528 000122A6 881D[FE880100] <1>      mov     [audio_cb_mode], bl
2529 000122AC 880D[FF880100] <1>      mov     [audio_srb], cl
2530                                <1>
2531                                <1>      ; 27/07/2020
2532                                <1>      ;mov     byte [audio_flag], 0 ; clear dma half buffer flag
2533                                <1>
2534                                <1>      ; 24/04/2017
2535 000122B2 803D[D1880100]03 <1>      cmp     byte [audio_device], 3 ; VT8233 (VT8237R)
2536 000122B9 7435      <1>      je      short sndc_init9
2537                                <1>      ;ja      short soundc_respond_err ; temporary (28/05/2017)
2538 000122BB 803D[D1880100]01 <1>      cmp     byte [audio_device], 1 ; SB 16
2539 000122C2 7510      <1>      jne     short sndc_init13
2540 000122C4 BB[73400100] <1>      mov     ebx, sb16_int_handler
2541                                <1>      ; Note: 'Sbinit' is at 'Start to Play' stage
2542                                <1>      ; 20/05/2017
2543 000122C9 66C705[0C890100]08- <1>      mov     word [audio_master_volume], 0808h ; 2/8
2544 000122D1 08      <1>
2545 000122D2 EB34      <1>      jmp     short sndc_init10
2546                                <1> sndc_init13:
2547 000122D4 803D[D1880100]02 <1>      ; 28/05/2017
2548                                <1>      cmp     byte [audio_device], 2 ; AC 97 (ICH)
2549                                <1>      ;jne     soundc_respond_err ; temporary (28/05/2017)
2550 000122DB 7405      <1>      ; 29/07/2022
2551                                <1>      je      short sndc_init16
2552                                <1> sndc_init15:
2553                                <1>      jmp     soundc_respond_err
2554 000122E2 E8D41E0000 <1>      jmp     sndc_init16:
2555                                <1>      call    ac97_codec_config
2556                                <1>      ;jc      soundc_respond_err ; codec error !
2557                                <1>      ; 29/07/2022
2558                                <1>      jc      short sndc_init15
2559 000122E9 BB[55440100] <1>      mov     ebx, ac97_int_handler
2560 000122EE EB18      <1>      jmp     short sndc_init10
2561                                <1>
2562                                <1> sndc_init9:
2563                                <1>      ;call    reset_codec
2564                                <1>      ;; eax = 1
2565                                <1>      ;call    codec_io_w16 ; w32
2566 000122F0 E8EC170000 <1>      call    init_codec ; 28/05/2017
2567                                <1>      ;jc      soundc_respond_err ; codec error !
2568                                <1>      ; 29/07/2022
2569 000122F5 72E6      <1>      jc      short sndc_init15
2570                                <1>
2571 000122F7 E8FA190000 <1>      call    channel_reset
2572                                <1>
2573                                <1>      ; setup the Codec (actually mixer registers)
2574 000122FC E823190000 <1>      call    codec_config ; unmute codec, set rates.
2575                                <1>      ;jc      soundc_respond_err ; codec error !
2576                                <1>      ; 29/07/2022
2577 00012301 72DA      <1>      jc      short sndc_init15
2578                                <1>
2579 00012303 BB[8A3C0100] <1>      mov     ebx, vt8233_int_handler
2580                                <1> sndc_init10:
2581                                <1>      ; 13/04/2017
2582 00012308 A0[D2880100] <1>      mov     al, [audio_intr] ; IRQ number
2583 0001230D E833FDFFFF <1>      call    set_dev_IRQ_service
2584                                <1>
2585                                <1>      ; SETUP (audio) INTERRUPT CALLBACK SERVICE
2586 00012312 8A1D[D2880100] <1>      mov     bl, [audio_intr] ; IRQ number
2587 00012318 8A3D[FE880100] <1>      mov     bh, [audio_cb_mode]
2588 0001231E FEC7      <1>      inc     bh ; 1 = Signal Response Byte method (fixed value)
2589                                <1>      ; 2 = Callback service method
2590                                <1>      ; 3 = Auto Increment S.R.B. method
2591 00012320 8A0D[FF880100] <1>      mov     cl, [audio_srb]
2592 00012326 8B15[00890100] <1>      mov     edx, [audio_cb_addr]
2593 0001232C A0[FD880100] <1>      mov     al, [audio_user]
2594                                <1>      ; 14/04/2017
2595 00012331 E83C050000 <1>      call    set_irq_callback_service
2596                                <1>      ; 16/04/2017
2597 00012336 A3[548E0100] <1>      mov     [u.r0], eax
2598                                <1>      ;jnc     sysret
2599 0001233B 7316      <1>      jnc     short sndc_init2 ; 21/04/2017
2600                                <1>      ;
2601 0001233D A3[C08E0100] <1>      mov     dword [u.error], eax
2602                                <1>
2603 00012342 A0[D2880100] <1>      mov     al, [audio_intr] ; IRQ number
2604 00012347 31DB      <1>      xor     ebx, ebx ; reset IRQ handler address
2605 00012349 E8F7FCFFFF <1>      call    set_dev_IRQ_service
2606                                <1>
2607 0001234E E90BA7FFFF <1>      jmp     error
2608                                <1>
2609                                <1> sndc_init2:
2610                                <1>      ; 21/04/2017
2611 00012353 8B0D[EC880100] <1>      mov     ecx, [audio_buff_size] ; audio buffer size
2612                                <1>      ; 05/06/2024
2613                                <1>      ;shl     ecx, 1 ; *2
2614                                <1>
2615                                <1>      ; 05/06/2024
2616 00012359 A0[D1880100] <1>      mov     al, [audio_device]
2617 0001235E 3C01      <1>      cmp     al, 1 ; SB16
2618 00012360 760A      <1>      jna     short sndc_init22
2619 00012362 80E1FE      <1>      and     cl, ~1 ; truncated for word alignment
2620 00012365 3C03      <1>      cmp     al, 3 ; VT8233
2621 00012367 7303      <1>      jnb     short sndc_init22
2622                                <1>      ; al = 2 ; AC'97
2623 00012369 80E1F8      <1>      and     cl, ~7 ; truncated for 8 byte (8x) alignment
2624                                <1> sndc_init22:
2625                                <1>      ; 05/06/2024
2626 0001236C 890D[F8880100] <1>      mov     [dma_hbuff_size], ecx ; DMA half buffer size
2627 00012372 D1E1      <1>      shl     ecx, 1 ; add ecx, ecx ; * 2

```

```

2628 <1>
2629 <1> ;;;
2630 <1> ; 04/06/2024
2631 00012374 81C1FF0F0000 <1> add ecx, 4095 ; PAGE_SIZE - 1
2632 0001237A 6681E100F0 <1> and cx, ~4095 ; ~PAGE_OFF
2633 <1> ; ecx = page border aligned DMA buffer size (required) (*)
2634 <1> ;;;
2635 <1>
2636 0001237F A1[F0880100] <1> mov eax, [audio_dma_buff]
2637 00012384 21C0 <1> and eax, eax
2638 00012386 741C <1> jz short sndc_init3 ; no need to compare dma buff size
2639 <1>
2640 00012388 8B15[F4880100] <1> mov edx, [audio_dmabuff_size] ; dma buffer size
2641 0001238E 39D1 <1> cmp ecx, edx
2642 00012390 742B <1> je short sndc_init5
2643 <1>
2644 <1> ; 04/12/2023
2645 00012392 3D00000500 <1> cmp eax, sb16_dma_buffer ; reserved buffer ?
2646 00012397 7409 <1> je short sndc_init20 ; it isn't an allocated mem buff
2647 <1>
2648 00012399 87D1 <1> xchg ecx, edx
2649 <1> ; 26/11/2023
2650 <1> ; round up (always -rounded up- page count is allocated)
2651 <1> ; ((so deallocation must be done for the rounded up value))
2652 <1> ; add ecx, PAGE_SIZE - 1 ; 4095
2653 <1> ; call deallocate_memory_block
2654 <1> ; 04/06/2024
2655 <1> ; call deallocate_memory_block_x
2656 <1> ; ; deallocate ((ecx+4095)>>12) pages
2657 0001239B E8B33BFFFF <1> call deallocate_memory_block
2658 <1>
2659 000123A0 87CA <1> xchg edx, ecx
2660 <1> sndc_init20:
2661 000123A2 31C0 <1> xor eax, eax
2662 <1> sndc_init3:
2663 <1> ; 05/06/2024
2664 000123A4 890D[F4880100] <1> mov [audio_dmabuff_size], ecx ; (*)
2665 <1> ; 12/05/2017
2666 000123AA 803D[D1880100]01 <1> cmp byte [audio_device], 1 ; SB 16
2667 000123B1 750B <1> jne short sndc_init4
2668 000123B3 C705[F0880100]0000- <1> mov dword [audio_dma_buff], sb16_dma_buffer
2669 000123BB 0500 <1>
2670 <1> ; 05/06/2024
2671 <1> ; mov dword [audio_dmabuff_size], 65536
2672 <1> ; xor eax, eax
2673 <1> ; mov [u.r0], eax ; 0 = no error, successful
2674 000123BD C3 <1> sndc_init5: ; 29/07/2022
2675 <1> retn
2676 <1>
2677 <1> sndc_init4:
2678 <1> ; 02/12/2023 - TRDOS 386 v2.0.7
2679 000123BE 81F900000100 <1> cmp ecx, 65536
2680 000123C4 7707 <1> ja short sndc_init21
2681 000123C6 B800000500 <1> mov eax, sb16_dma_buffer ; use already reserved buffer
2682 000123CB EB0C <1> jmp short sndc_init17
2683 <1>
2684 <1> sndc_init21:
2685 <1> ; EAX = Beginning address (physical)
2686 <1> ; EAX = 0 -> Allocate mem block from the 1st proper aperture
2687 000123CD E86739FFFF <1> ; ECX = Number of bytes to be allocated (>0)
2688 <1> call allocate_memory_block
2689 <1> ; jc sound_buff_error
2690 000123D2 7305 <1> ; 29/07/2022
2691 000123D4 E988FDFFFF <1> jnc short sndc_init17
2692 <1> jmp sound_buff_error
2693 <1>
2694 <1> sndc_init17: ; 29/07/2022
2695 000123D9 A3[F0880100] <1> ; set dma buffer address and size parameters
2696 <1> mov [audio_dma_buff], eax ; dma buffer address
2697 <1> ; 05/06/2024
2698 <1> ; mov [audio_dmabuff_size], ecx ; dma buffer size
2699 <1>
2700 <1> ;;;
2701 000123DE 8B0D[F8880100] <1> ; 05/06/2024
2702 <1> mov ecx, [dma_hbuff_size] ; DMA half (1-2) buffer size
2703 <1> ; 04/06/2024
2704 <1> ; mov ecx, [audio_buff_size] ; audio buffer size in bytes
2705 <1> ;;;
2706 <1> ; ; EAX = Beginning (physical) addr of the allocated mem block
2707 <1> ; ; ECX = Num of allocated bytes (rounded up to page borders)
2708 <1> ; cmp byte [audio_pci], 0 ; AC97 audio controller ?
2709 <1> ; ja short sndc_init4
2710 <1> ;
2711 <1> ; ; Sound Blaster 16 uses classic DMA
2712 <1> ; mov edx, eax
2713 <1> ; add edx, ecx
2714 <1> ; cmp edx, 1000000h ; 1st 16 MB
2715 <1> ; jna short sndc_init4
2716 <1> ;
2717 <1> ; error !
2718 <1> ; restore Memory Allocation Table Content
2719 <1> ; EAX = Beginning address (physical)
2720 <1> ; ECX = Number of bytes to be deallocated
2721 <1> ; call deallocate_memory_block
2722 <1> ; reset dma buffer address and size parameters
2723 <1> ; xor eax, eax ; 0
2724 <1> ; mov [audio_dma_buff], eax ; 0
2725 <1> ; mov [audio_dmabuff_size], ecx ; 0
2726 <1> ; jmp sound_buff_error
2727 <1> ;
2728 <1> ; sndc_init4:
2729 000123E4 803D[D1880100]03 <1> cmp byte [audio_device], 3
2730 <1> ; jne short sndc_init5
2731 <1> ; 29/07/2022
2732 000123EB 7505 <1> jne short sndc_init18 ; 28/05/2017
2733 <1>
2734 <1> ; 29/07/2022
2735 <1> ; call set_vt8233_bd1
2736 <1> ; sndc_init5:
2737 <1> ; sub eax, eax ; 0
2738 <1> ; mov [u.r0], eax ; 0 = no error, successful
2739 <1> ; retn
2740 <1>
2741 <1> ; 05/06/2024
2742 <1> ;;;
2743 <1> ; 04/06/2024 (use truncated buffer size for BDL setup)
2744 <1> ; NOTE: Round up adds spurious bytes (noise) to the DMA buff;
2745 <1> ; so, round down the size is better than the round up.
2746 <1> ; ((BDL/SGD feature needs word aligned buffer address))
2747 <1> ;
2748 <1> ; mov ecx, [audio_buff_size] ; audio buffer size in bytes
2749 <1> ; and cl, ~1
2750 <1> ; 05/06/2024

```



```

2751      <1>      ;mov     [dma_hbuff_size], ecx
2752      <1>      ; ecx = DMA half buffer size as word aligned
2753      <1>      ;      (in fact, half buffer is one of the two DMA buffers)
2754      <1>      ;;;
2755      <1>
2756 000123ED E93E190000      <1>      jmp     set_vt8233_bdl
2757      <1>
2758      <1> sndc_init18:
2759      <1>      ;call    set_ac97_bdl
2760      <1>      ;;;jmp    short sndc_init5
2761      <1>      ;retn
2762      <1>
2763      <1>      ; 05/06/2024
2764      <1>      ;;;
2765      <1>      ;;;
2766      <1>      ; 04/06/2024 (use truncated buffer size for BDL setup)
2767      <1>      ; NOTE: Round up adds spurious bytes (noise) to the DMA buff;
2768      <1>      ; so, round down the size is better than the round up.
2769      <1>      ; ((BDL feature needs 8 byte aligned buffer address))
2770      <1>      ;
2771      <1>      ;mov     ecx, [audio_buff_size] ; audio buffer size in bytes
2772      <1>      ;and     cl, ~7
2773      <1>      ; 05/06/2024
2774      <1>      ;mov     [dma_hbuff_size], ecx
2775      <1>      ; ecx = DMA half buffer size as truncated for 8 byte alignment
2776      <1>      ;      (in fact, half buffer is one of the two DMA buffers)
2777      <1>      ;;;
2778      <1>
2779      <1>      ; 29/07/2022
2780 000123F2 E96A1F0000      <1>      jmp     set_ac97_bdl
2781      <1>
2782      <1> sound_play:
2783      <1>      ; FUNCTION = 4
2784      <1>      ; b1 = Mode
2785      <1>      ; bit 0 = mono/stereo (1 = stereo)
2786      <1>      ; bit 1 = 8 bit / 16 bit (1 = 16 bit)
2787      <1>      ; cx = Sampling Rate (Hz)
2788      <1>
2789      <1>      ; 13/06/2017
2790      <1>      ; Note: Even if Mode bits are not 11b,
2791      <1>      ; AC'97 Audio Controller (&Codec)
2792      <1>      ; will play audio samples as 16 bit, stereo
2793      <1>      ; samples.
2794      <1>      ; (Program must fill the audio buffer
2795      <1>      ; as required; 8 bit samples must be converted
2796      <1>      ; to 16 bit samples and mono samples must be
2797      <1>      ; converted to stereo samples...)
2798      <1>
2799      <1>      ; 19/12/2024 (BugFix)
2800      <1>      ; 05/06/2024
2801      <1>      ; 04/06/2024
2802      <1>      ; 30/07/2022
2803      <1>      ; 28/07/2020
2804      <1>      ; 27/07/2020
2805      <1>      ; 28/05/2017
2806      <1>      ; 15/05/2017, 20/05/2017
2807      <1>      ; 21/04/2017, 24/04/2017
2808      <1>      ; ... device check at first
2809 000123F7 A0D1880100      <1>      mov     al, [audio_device]
2810 000123FC 08C0            <1>      or      al, al ; 0 ; pc speaker or invalid
2811      <1>      ;jz      beeper_gfx ; 'video.s' ; temporary
2812      <1>      ; 30/07/2022
2813 000123FE 7505            <1>      jnz     short snd_play_7
2814 00012400 E9C9FFFEFF      <1>      jmp     beeper_gfx
2815      <1>
2816      <1> snd_play_7:
2817      <1>      ; cmp     al, 3 ; VIA VT 8237R (vt8233)
2818      <1>      ; je      short snd_play_1
2819      <1>      ; cmp     al, 1 ; SB 16
2820      <1>      ; jne     soundc_dev_err ; temporary !
2821      <1> ;snd_play_0:
2822      <1>      ; ... buffer & (buffer) owner check at second
2823 00012405 833D[E4880100]00      <1>      cmp     dword [audio_buffer], 0
2824      <1>      ;jna     sound_buff_error
2825      <1>      ; 30/07/2022
2826 0001240C 763C            <1>      jna     short snd_play_4 ; jmp sound_buff_error
2827 0001240E A0[A58E0100]      <1>      mov     al, [u.uno]
2828 00012413 3A05[FD880100]      <1>      cmp     al, [audio_user]
2829      <1>      ;jne     sndc_owner_error
2830      <1>      ; 30/07/2022
2831 00012419 7405            <1>      je      short snd_play_3
2832 0001241B E967FEFFFF      <1>      jmp     sndc_owner_error
2833      <1> snd_play_3:
2834 00012420 66890D[06890100]      <1>      mov     [audio_freq], cx ; sample frequency (Hertz)
2835 00012427 88D8            <1>      mov     al, b1
2836 00012429 2401            <1>      and     al, 1 ; mono/stereo (1= stereo)
2837 0001242B FEC0            <1>      inc     al ; channels
2838 0001242D A2[05890100]      <1>      mov     [audio_stmo], al ; sound channels (1 or 2)
2839 00012432 B008            <1>      mov     al, 8
2840 00012434 F6C302          <1>      test    b1, 2 ; bits per sample (1= 16 bit)
2841 00012437 7402            <1>      jz      short snd_play_bps
2842 00012439 D0E0            <1>      shl     al, 1
2843      <1> snd_play_bps:
2844 0001243B A2[04890100]      <1>      mov     [audio_bps], al
2845      <1>
2846      <1>      ; Transfer ring 3 (user's) audio buffer content to dma buffer
2847 00012440 8B3D[F0880100]      <1>      mov     edi, [audio_dma_buff] ; dma buffer (ring 0)
2848 00012446 09FF            <1>      or      edi, edi
2849      <1>      ;jz      sound_buff_error
2850      <1>      ; 30/07/2022
2851 00012448 7505            <1>      jnz     short snd_play_5
2852      <1> snd_play_4:
2853 0001244A E912FDFFFF      <1>      jmp     sound_buff_error
2854      <1> snd_play_5:
2855      <1>      ; 27/07/2020
2856 0001244F 8B35[E8880100]      <1>      mov     esi, [audio_p_buffer] ; physical address (ring 3)
2857      <1>      ;mov     ecx, [audio_buff_size] ; 15/05/2017
2858      <1>
2859      <1>      ; 04/06/2024
2860      <1> %if 0
2861      <1>      mov     ecx, [audio_dmabuff_size] ; 27/07/2020
2862      <1>      ;or      ecx, ecx
2863      <1>      ;jz      sound_buff_error
2864      <1>      ; 28/07/2020
2865      <1>      shr     ecx, 1 ; dma half buffer size
2866      <1> %else
2867      <1>      ; 05/06/2024
2868 00012455 8B0D[F8880100]      <1>      mov     ecx, [dma_hbuff_size] ; DMA half buffer size
2869      <1>
2870      <1>      ; 04/06/2024
2871      <1>      ;mov     ecx, [audio_buff_size]
2872      <1>      ; 19/12/2024 (BugFix)
2873 0001245B A0[D1880100]      <1>      mov     al, [audio_device]
2874      <1>      ;

```

```

2875      <1>      ;cmp     al, 1 ; Sound Blaster 16
2876      <1>      ;je      short snd_play_8
2877      <1>      ;and     cl, ~1
2878      <1>      ;cmp     al, 3 ; VT8233 (VT8237R)
2879      <1>      ;je      short snd_play_8
2880      <1>      ; AC'97
2881      <1>      ;and     cl, ~7
2882      <1>      snd_play_8:
2883      <1>      %endif
2884      <1>
2885      00012460 8035[FC880100]01      <1>      xor     byte [audio_flag], 1 ; 0 -> 1, 1 -> 0
2886      00012467 7502                <1>      jnz     short snd_play_0 ; [audio_flag] = 1
2887      <1>      ; fill dma half buffer 1
2888      <1>      ; [audio_flag] = 0
2889      <1>
2890      <1>      ; fill dma half buffer 2
2891      00012469 01CF                <1>      add     edi, ecx
2892      <1>
2893      <1>      snd_play_0:
2894      <1>      ; 05/06/2024
2895      <1>      ; rep movsb
2896      <1>      shr     ecx, 2 ; convert byte count to dword count
2897      <1>      ; rep movsd ;
2898      0001246B F3A4                <1>      rep     movsb ; SB16, AC97, VT8233
2899      <1>
2900      <1>      ; here, if [audio_flag] = 0, interrupt handler will update
2901      <1>      ; dma half buffer 2
2902      <1>      ; (user's audio buffer data will be
2903      <1>      ; copied into dma half buffer 2)
2904      <1>      ; 20/05/2017
2905      <1>      ; mov     byte [audio_flag], 1 ; next half (on next time)
2906      <1>
2907      <1>      ; 24/04/2017
2908      <1>      ; mov     al, [audio_device]
2909      <1>      ; 04/06/2024
2910      <1>      ; al = [audio_device]
2911      <1>
2912      0001246D 3C03                <1>      cmp     al, 3 ; VT8233 (VT8237R)
2913      0001246F 7409                <1>      je      short snd_play_1
2914      00012471 3C01                <1>      cmp     al, 1 ; Sound Blaster 16
2915      00012473 750A                <1>      jne     short snd_play_2 ; 28/05/2017
2916      <1>
2917      <1>      ; 30/07/2022
2918      <1>      ; call    SbInit_play
2919      <1>      ; jc      soundc_respond_err
2920      <1>      ; retn
2921      <1>      ; 30/07/2022
2922      <1>      ; jnc     short snd_play_6 ; retn
2923      <1>      ; jmp     soundc_respond_err
2924      <1>
2925      <1>      ; 30/07/2022
2926      00012475 E9A71A0000          <1>      jmp     SbInit_play ; sb16_start_play
2927      <1>
2928      <1>      snd_play_1:
2929      <1>      ; call    vt8233_start_play
2930      <1>      ; retn
2931      <1>      ; 30/07/2022
2932      0001247A E9E5180000          <1>      jmp     vt8233_start_play
2933      <1>
2934      <1>      snd_play_2:
2935      <1>      ; 28/05/2017
2936      <1>      ; cmp     al, 2 ; AC'97
2937      <1>      ; jne     short snd_play_3
2938      <1>
2939      <1>      ; call    ac97_start_play
2940      <1>      ; retn
2941      <1>      ; 30/07/2022
2942      0001247F E90E1F0000          <1>      jmp     ac97_start_play
2943      <1>
2944      <1>      ; snd_play_3:
2945      <1>      ; call    hda_start_play
2946      <1>      ; retn
2947      <1>      ; 30/07/2022
2948      <1>      ; jmp     hda_start_play
2949      <1>
2950      <1>      ; 04/06/2024
2951      <1>      ; snd_play_6:
2952      <1>      ; 30/07/2022
2953      <1>      ; retn
2954      <1>
2955      <1>      sound_pause:
2956      <1>      ; FUNCTION = 5
2957      <1>      ; Pause
2958      <1>      ; 28/05/2017
2959      <1>      ; 24/04/2017
2960      <1>      ; 22/04/2017
2961      00012484 E82B030000          <1>      call    snd_dev_check
2962      00012489 7277                <1>      jc      short snd_nothing ; temporary.
2963      0001248B E831030000          <1>      call    snd_buf_check
2964      00012490 7270                <1>      jc      short snd_nothing ; temporary.
2965      00012492 A0[D1880100]        <1>      mov     al, [audio_device]
2966      00012497 3C03                <1>      cmp     al, 3 ; VIA VT 8237R (vt8233)
2967      00012499 7409                <1>      je      short snd_pause_1
2968      0001249B 3C01                <1>      cmp     al, 1 ; Sound Blaster 16
2969      0001249D 750A                <1>      jne     short snd_pause_2 ; 28/05/2017
2970      0001249F E9501C0000          <1>      jmp     sb16_pause
2971      <1>      snd_pause_1:
2972      000124A4 E970190000          <1>      jmp     vt8233_pause
2973      <1>      snd_pause_2:
2974      <1>      ; 28/05/2017
2975      <1>      ; cmp     al, 2 ; AC'97
2976      <1>      ; jne     short snd_nothing ; temporary.
2977      000124A9 E9E01F0000          <1>      jmp     ac97_pause
2978      <1>
2979      <1>      sound_continue:
2980      <1>      ; FUNCTION = 6
2981      <1>      ; Continue to play
2982      <1>      ; 28/05/2017
2983      <1>      ; 22/04/2017
2984      000124AE E801030000          <1>      call    snd_dev_check
2985      000124B3 724D                <1>      jc      short snd_nothing ; temporary.
2986      000124B5 E807030000          <1>      call    snd_buf_check
2987      000124BA 7246                <1>      jc      short snd_nothing ; temporary.
2988      000124BC A0[D1880100]        <1>      mov     al, [audio_device]
2989      000124C1 3C03                <1>      cmp     al, 3 ; VIA VT 8237R (vt8233)
2990      000124C3 7409                <1>      je      short snd_cont_1
2991      000124C5 3C01                <1>      cmp     al, 1 ; Sound Blaster 16
2992      000124C7 750A                <1>      jne     short snd_cont_2 ; 28/05/2017
2993      000124C9 E9481C0000          <1>      jmp     sb16_continue
2994      <1>      snd_cont_1:
2995      000124CE E9F8180000          <1>      jmp     vt8233_play
2996      <1>      snd_cont_2:
2997      <1>      ; 28/05/2017
2998      <1>      ; cmp     al, 2 ; AC'97

```

```

2999      <1>      ;;jne short snd_nothing ; temporary.
3000      <1>      ; 30/07/2022
3001      <1>      ;;jne short snd_cont_3
3002 000124D3 E9F71E0000 <1>      jmp ac97_play
3003      <1>      ;snd_cont_3:
3004      <1>      ;jmp hda_play
3005      <1>
3006      <1>      sound_stop:
3007      <1>      ; FUNCTION = 7
3008      <1>      ; Stop playing
3009      <1>      ; 30/07/2022
3010      <1>      ; 28/05/2017
3011      <1>      ; 24/05/2017
3012      <1>      ; 21/04/2017, 22/04/2017, 24/04/2017
3013 000124D8 E8D7020000 <1>      call snd_dev_check
3014 000124DD 7223 <1>      jc short snd_nothing ; temporary.
3015      <1>      ;call snd_buf_check
3016 000124DF E8E6020000 <1>      call snd_user_check ; 24/05/2017
3017 000124E4 721C <1>      jc short snd_nothing ; temporary.
3018      <1>
3019 000124E6 A0[D1880100] <1>      mov al, [audio_device]
3020 000124EB 3C03 <1>      cmp al, 3 ; VIA VT 8237R (vt8233)
3021      <1>      ;je vt8233_stop
3022      <1>      ; 28/05/2017
3023      <1>      ;ja short snd_nothing
3024      <1>      ; 30/07/2022
3025 000124ED 7505 <1>      jne short snd_stop_1
3026 000124EF E933180000 <1>      jmp vt8233_stop
3027      <1>      snd_stop_1:
3028 000124F4 3C01 <1>      cmp al, 1 ; Sound Blaster 16
3029      <1>      ;je sb16_stop
3030      <1>      ; 30/07/2022
3031 000124F6 7505 <1>      jne short snd_stop_2
3032 000124F8 E93B1C0000 <1>      jmp sb16_stop
3033      <1>      snd_stop_2:
3034      <1>      ;cmp al, 2
3035      <1>      ;je short ac97_stop
3036      <1>      ; 30/07/2022
3037      <1>      ;jne short snd_stop_3
3038 000124FD E9831F0000 <1>      jmp ac97_stop ; temporary.
3039      <1>      ;snd_stop_3:
3040      <1>      ;jmp hda_stop
3041      <1>
3042      <1>      sndc_cancel_ok:
3043      <1>      ; 30/07/2022
3044      <1>      sndc_reset_ok:
3045      <1>      ; 30/07/2022
3046      <1>      snd_nothing:
3047      <1>      ; 21/04/2017
3048 00012502 C3 <1>      retn
3049      <1>
3050      <1>      soundc_reset:
3051      <1>      ; FUNCTION = 8
3052      <1>      ; Reset Audio Controller
3053      <1>      ; 30/07/2022
3054      <1>      ; 28/05/2017
3055      <1>      ; 22/04/2017
3056 00012503 E8AC020000 <1>      call snd_dev_check
3057 00012508 72F8 <1>      jc short snd_nothing ; temporary.
3058 0001250A E8B2020000 <1>      call snd_buf_check
3059 0001250F 72F1 <1>      jc short snd_nothing ; temporary.
3060      <1>
3061 00012511 A0[D1880100] <1>      mov al, [audio_device]
3062      <1>      ; 30/07/2022
3063 00012516 3C03 <1>      cmp al, 3 ; VIA VT 8237R (vt8233)
3064 00012518 7207 <1>      jb short sndc_reset_1
3065      <1>      ;je vt8233_reset
3066 0001251A 77E6 <1>      ja short snd_nothing ; temporary.
3067      <1>      ;ja hda_reset
3068      <1>      ;ja short sndc_reset_3
3069      <1>      ; 30/07/2022
3070 0001251C E904190000 <1>      jmp vt8233_reset
3071      <1>      sndc_reset_1:
3072 00012521 3C01 <1>      cmp al, 1 ; Sound Blaster 16
3073      <1>      ;jne ac97_reset
3074      <1>      ; 30/07/2022
3075 00012523 750C <1>      jne short sndc_reset_2
3076 00012525 E85F1C0000 <1>      call sb16_reset
3077      <1>      ;jc soundc_respond_err
3078      <1>      ;retn
3079      <1>      ; 30/07/2022
3080 0001252A 73D6 <1>      jnc short sndc_reset_ok
3081 0001252C E90BFCFFFF <1>      jmp soundc_respond_err
3082      <1>      sndc_reset_2:
3083      <1>      ; 30/07/2022
3084      <1>      ;cmp al, 2
3085      <1>      ;ja short sndc_reset_3
3086 00012531 E914200000 <1>      jmp ac97_reset
3087      <1>      ;sndc_reset_3:
3088      <1>      ;jmp hda_reset
3089      <1>
3090      <1>      soundc_cancel:
3091      <1>      ; FUNCTION = 9
3092      <1>      ; Cancel audio callback service
3093      <1>      ; 30/07/2022
3094      <1>      ; 22/04/2017
3095 00012536 A0[FD880100] <1>      mov al, [audio_user]
3096 0001253B 3A05[A58E0100] <1>      cmp al, [u.uno]
3097 00012541 75BF <1>      jne short snd_nothing
3098      <1>      ; RESET (audio) INTERRUPT CALLBACK SERVICE
3099 00012543 8A1D[D2880100] <1>      mov bl, [audio_intr] ; IRQ number
3100 00012549 A0[A58E0100] <1>      mov al, [u.uno]
3101 0001254E 28FF <1>      sub bh, bh ; 0 ; unlink IRQ from user service
3102 00012550 E81D030000 <1>      call set_irq_callback_service
3103      <1>      ;jc sndc_perm_error ; 'permission denied' error
3104      <1>      ;retn
3105      <1>      ; 30/07/2022
3106 00012555 73AB <1>      jnc short sndc_cancel_ok
3107 00012557 E930FDFFFF <1>      jmp sndc_perm_error
3108      <1>
3109      <1>      sound_dalloc:
3110      <1>      ; FUNCTION = 10
3111      <1>      ; Deallocate (ring 3) audio buffer
3112      <1>      ; 22/04/2017
3113 0001255C A0[FD880100] <1>      mov al, [audio_user]
3114 00012561 3A05[A58E0100] <1>      cmp al, [u.uno]
3115 00012567 7599 <1>      jne short snd_nothing
3116 00012569 8B1D[E4880100] <1>      mov ebx, [audio_buffer]
3117      <1>      ;or ebx, ebx
3118      <1>      ;jz short snd_nothing
3119 0001256F 8B0D[EC880100] <1>      mov ecx, [audio_buff_size]
3120 00012575 E8173BFFFF <1>      call deallocate_user_pages
3121 0001257A 31C0 <1>      xor eax, eax
3122 0001257C A3[E4880100] <1>      mov [audio_buffer], eax ; 0

```

```

3123 00012581 A2[FD880100] <1> mov [audio_user], al ; 0
3124 <1> ;sndc_cancel_ok:
3125 00012586 C3 <1> retn
3126 <1>
3127 <1> sound_volume:
3128 <1> ; FUNCTION = 11
3129 <1> ; Set sound volume level
3130 <1> ; 23/05/2024
3131 <1> ; 30/07/2022
3132 <1> ; 28/05/2017
3133 <1> ; 20/05/2017
3134 <1> ; 22/04/2017, 24/04/2017
3135 <1> ; b1 = component
3136 <1> ; 0 = master/playback/lineout volume
3137 <1> ; 1 = PCM out volume ; 23/05/2024
3138 <1> ; cl = left channel volume level (0 to 31)
3139 <1> ; ch = right channel volume level (0 to 31)
3140 <1>
3141 00012587 80FB80 <1> cmp b1, 80h
3142 0001258A 720A <1> jb short snd_vol_1
3143 <1> ;ja snd_nothing ; temporary.
3144 <1> ; 30/07/2022
3145 0001258C 7707 <1> ja short snd_vol_0
3146 <1> ; Set volume level for next play (BL >= 80h)
3147 0001258E 66890D[0C890100] <1> mov [audio_master_volume], cx
3148 <1> snd_vol_0:
3149 00012595 C3 <1> retn
3150 <1> snd_vol_1:
3151 <1> ; set volume level immediate (BL < 80h)
3152 <1> ;cmp b1, 0
3153 <1> ;ja snd_nothing ; temporary.
3154 <1> ; 30/07/2022
3155 <1> ;ja short snd_vol_0
3156 <1> ; 23/05/2024
3157 00012596 80FB01 <1> cmp b1, 1
3158 00012599 77FA <1> ja short snd_vol_0 ; temporary
3159 <1>
3160 0001259B E814020000 <1> call snd_dev_check
3161 <1> ;jc snd_nothing ; temporary.
3162 <1> ; 30/07/2022
3163 000125A0 72F3 <1> jc short snd_vol_0
3164 000125A2 E81A020000 <1> call snd_buf_check
3165 <1> ;jc snd_nothing ; temporary.
3166 <1> ; 30/07/2022
3167 000125A7 72EC <1> jc short snd_vol_0
3168 <1>
3169 000125A9 A0[D1880100] <1> mov al, [audio_device]
3170 000125AE 3C03 <1> cmp al, 3 ; VIA VT 8237R (vt8233)
3171 <1> ;je vt8233_volume
3172 <1> ; 30/07/2022
3173 000125B0 7207 <1> jb short snd_vol_2
3174 <1> ; 28/05/2017
3175 <1> ;ja snd_nothing ; temporary.
3176 <1> ; 30/07/2022
3177 000125B2 77E1 <1> ja short snd_vol_0
3178 <1> ;ja hda_volume
3179 <1> ; 30/07/2022
3180 000125B4 E985180000 <1> jmp vt8233_volume
3181 <1> snd_vol_2:
3182 <1> ; Sound Blaster 16
3183 000125B9 3C01 <1> cmp al, 1 ; SB 16
3184 <1> ;je sb16_volume
3185 <1> ; 30/07/2022
3186 000125BB 7705 <1> ja short snd_vol_3
3187 000125BD E9011B0000 <1> jmp sb16_volume
3188 <1> snd_vol_3:
3189 <1> ; 30/07/2022
3190 <1> ;cmp al, 2
3191 <1> ;ja short snd_vol_4
3192 000125C2 E9541E0000 <1> jmp ac97_volume
3193 <1> ;snd_vol_4:
3194 <1> ;jmp hda_volume
3195 <1>
3196 <1> soundc_disable:
3197 <1> ; FUNCTION = 12
3198 <1> ; Disable audio device (and unlink DMA memory)
3199 <1> ; 23/08/2024
3200 <1> ; 04/06/2024
3201 <1> ; 30/07/2022
3202 <1> ; 28/05/2017
3203 <1> ; 24/05/2017
3204 <1> ; 22/04/2017
3205 000125C7 E8E8010000 <1> call snd_dev_check
3206 <1> ;jc soundc_dev_err ; temporary.
3207 <1> ; 30/07/2022
3208 000125CC 7305 <1> jnc short snd_disable_4
3209 000125CE E962FBFFFF <1> jmp soundc_dev_err
3210 <1> snd_disable_4:
3211 <1> ;call snd_buf_check
3212 <1> ;jc sndc_owner_error ; temporary.
3213 <1> ; 30/07/2022
3214 <1> ;jnc short snd_disable_5
3215 <1> ;jmp sndc_owner_error
3216 <1> ;snd_disable_5:
3217 000125D3 A0[D1880100] <1> mov al, [audio_device]
3218 000125D8 3C03 <1> cmp al, 3 ; VIA VT 8237R (vt8233)
3219 000125DA 7414 <1> je short snd_disable_1
3220 <1> ;ja snd_nothing ; temporary.
3221 <1> ; 30/07/2022
3222 000125DC 7766 <1> ja short snd_disable_3 ; retn
3223 000125DE 3C01 <1> cmp al, 1 ; Sound Blaster 16
3224 000125E0 7507 <1> jne short snd_disable_0
3225 000125E2 E8511B0000 <1> call sb16_stop
3226 000125E7 EB0C <1> jmp short snd_disable_2
3227 <1> snd_disable_0:
3228 000125E9 E8971E0000 <1> call ac97_stop
3229 000125EE EB05 <1> jmp short snd_disable_2
3230 <1> snd_disable_1:
3231 000125F0 E832170000 <1> call vt8233_stop
3232 <1> snd_disable_2:
3233 000125F5 A0[D2880100] <1> mov al, [audio_intr]
3234 <1>
3235 <1> ; 15/01/2025
3236 000125FA 20C0 <1> and al, al
3237 000125FC 740E <1> jz short snd_disable_5
3238 <1>
3239 000125FE 29DB <1> sub ebx, ebx ; 0 = reset
3240 00012600 E840FAFFFF <1> call set_dev_IRQ_service
3241 <1>
3242 <1> ;mov al, [audio_intr]
3243 00012605 28E4 <1> sub ah, ah ; 0 = reset
3244 00012607 E800F8FFFF <1> call set_hardware_int_vector
3245 <1>
3246 <1> snd_disable_5: ; 15/01/2025

```

```

3247 0001260C 31C0      <1>    xor     eax, eax
3248 0001260E A2[D1880100]  <1>    mov     byte [audio_device], al
3249 00012613 A2[D2880100]  <1>    mov     byte [audio_intr], al
3250 00012618 8705[F0880100] <1>    xchg     eax, [audio_dma_buff]
3251                                <1>
3252                                <1>    ; 15/01/2025
3253 0001261E 09C0      <1>    or      eax, eax
3254 00012620 7422      <1>    jz      short snd_disable_3
3255                                <1>
3256                                <1>    ; 24/05/2017
3257                                <1>    ;or      eax, eax
3258                                <1>    ;jz      short snd_disable_3
3259                                <1>    ;cmp     eax, sb16_dma_buffer ; default DMA buffer
3260                                <1>    ;je      short snd_disable_3
3261 00012622 803D[D0880100]00 <1>    cmp     byte [audio_pci], 0 ; AC97 audio controller ?
3262 00012629 7619      <1>    jna     short snd_disable_3
3263 0001262B C605[D0880100]00 <1>    mov     byte [audio_pci], 0
3264                                <1>
3265                                <1>    ; 23/08/2024 - bugfix
3266 00012632 3D00000500 <1>    cmp     eax, sb16_dma_buffer ; reserved buffer ?
3267 00012637 740B      <1>    je      short snd_disable_3 ; it isn't an allocated mem buff
3268                                <1>
3269                                <1>    ;sub     ecx, ecx
3270                                <1>    ;xchg     ecx, [audio_dmabuff_size]
3271 00012639 8B0D[F4880100] <1>    mov     ecx, [audio_dmabuff_size]
3272                                <1>    ; 26/11/2023
3273                                <1>    ; round up (always -rounded up- page count is allocated)
3274                                <1>    ; ((so deallocation must be done for the rounded up value))
3275                                <1>    ;add     ecx, PAGE_SIZE - 1 ; 4095
3276                                <1>    ;call    deallocate_memory_block
3277                                <1>    ; 04/06/2024
3278                                <1>    ;call    deallocate_memory_block_x
3279                                <1>    ; deallocate ((ecx+4095)>>12) pages
3280 0001263F E80F39FFFF <1>    call    deallocate_memory_block
3281                                <1>    snd_disable_3:
3282 00012644 C3          <1>    retn
3283                                <1>
3284                                <1>    sound_dma_map:
3285                                <1>    ; FUNCTION = 13
3286                                <1>    ; Map audio dma buff addr to user's buffer addr
3287                                <1>    ; 30/07/2022
3288                                <1>    ; 12/05/2017
3289 00012645 21C9      <1>    and     ecx, ecx
3290                                <1>    ;jz      sound_buff_error
3291                                <1>    ; 30/07/2022
3292 00012647 7505      <1>    jnz     short snd_dma_map_3
3293 00012649 E913FBFFFF <1>    jmp     sound_buff_error
3294                                <1>    snd_dma_map_3:
3295 0001264E 803D[D1880100]01 <1>    cmp     byte [audio_device], 1
3296 00012655 722A      <1>    jb      short snd_dma_map_1
3297                                <1>    snd_dma_map_0:
3298 00012657 A1[F0880100] <1>    mov     eax, [audio_dma_buff]
3299 0001265C 21C0      <1>    and     eax, eax
3300 0001265E 7421      <1>    jz      short snd_dma_map_1
3301                                <1>
3302 00012660 8A1D[FD880100] <1>    mov     bl, [audio_user]
3303 00012666 08DB      <1>    or      bl, bl
3304 00012668 7417      <1>    jz      short snd_dma_map_1
3305 0001266A 3A1D[A58E0100] <1>    cmp     bl, [u.uno]
3306                                <1>    ;jne     sndc_owner_error
3307                                <1>    ; 30/07/2022
3308 00012670 7405      <1>    je      short snd_dma_map_4
3309 00012672 E910FCFFFF <1>    jmp     sndc_owner_error
3310                                <1>    snd_dma_map_4:
3311 00012677 8B1D[F4880100] <1>    mov     ebx, [audio_dmabuff_size]
3312 0001267D 21DB      <1>    and     ebx, ebx
3313 0001267F 750A      <1>    jnz     short snd_dma_map_2
3314                                <1>    snd_dma_map_1:
3315 00012681 B800000500 <1>    mov     eax, sb16_dma_buffer
3316 00012686 B800000100 <1>    mov     ebx, 65536
3317                                <1>    snd_dma_map_2:
3318 0001268B 81C1FF0F0000 <1>    add     ecx, PAGE_SIZE-1 ; 4095
3319 00012691 6681E100F0 <1>    and     cx, ~PAGE_OFF ; not 4095
3320 00012696 39D9      <1>    cmp     ecx, ebx
3321                                <1>    ;ja      sound_buff_error
3322                                <1>    ; 30/07/2022
3323 00012698 7605      <1>    jna     short snd_dma_map_6
3324                                <1>    snd_dma_map_5:
3325 0001269A E9C2FAFFFF <1>    jmp     sound_buff_error
3326                                <1>    snd_dma_map_6:
3327 0001269F 50          <1>    push    eax
3328 000126A0 89D3      <1>    mov     ebx, edx
3329 000126A2 C1E90C      <1>    shr     ecx, 12 ; byte count to page count
3330                                <1>    ; eax = physical address of (audio) dma buffer
3331                                <1>    ; ebx = virtual address of (audio) dma buffer (user's pgdir)
3332                                <1>    ; ecx = page count (>0)
3333 000126A5 E81939FFFF <1>    call    direct_memory_access
3334 000126AA 58          <1>    pop     eax
3335                                <1>    ;jc      sound_buff_error
3336                                <1>    ; 30/07/2022
3337 000126AB 72ED      <1>    jc      short snd_dma_map_5
3338 000126AD A3[548E0100] <1>    mov     [u.r0], eax
3339 000126B2 C3          <1>    retn
3340                                <1>
3341                                <1>    soundc_info:
3342                                <1>    ; FUNCTION = 14
3343                                <1>    ; Get Audio Controller Info
3344                                <1>    ; 19/11/2023
3345                                <1>    ; 30/07/2022
3346                                <1>    ; 10/06/2017
3347                                <1>    ; 05/06/2017
3348                                <1>
3349                                <1>    ;and     bl, bl ; 0
3350                                <1>    ;jz      short sndc_info_0
3351                                <1>
3352                                <1>    ; 19/11/2023
3353 000126B3 80FB01 <1>    cmp     bl, 1
3354 000126B6 7609      <1>    jna     short sndc_info_0
3355                                <1>
3356                                <1>    ; invalid parameter !
3357                                <1>    ; 30/07/2022
3358                                <1>    ;mov     eax, ERR_INV_PARAMETER ; 23
3359                                <1>    ;sndc_inf_error:
3360                                <1>    ; mov     [u.r0], eax
3361                                <1>    ; mov     [u.error], eax
3362                                <1>    ; jmp     error
3363                                <1>    ; 30/07/2022
3364 000126B8 29C0      <1>    sub     eax, eax
3365 000126BA B017      <1>    mov     al, ERR_INV_PARAMETER ; 23
3366 000126BC E980FAFFFF <1>    jmp     sysaudio_err
3367                                <1>
3368                                <1>    sndc_info_0:
3369 000126C1 E8EE000000 <1>    call    snd_dev_check
3370                                <1>    ;jc      soundc_dev_err

```

```

3371      ; 30/07/2022
3372 000126C6 7305      jnc     short sndc_info_3
3373      snd_data_dev_err:
3374 000126C8 E968FAFFFF      jmp     soundc_dev_err
3375      sndc_info_3:
3376      ; 19/11/2023
3377      ; cmp     bl, 1
3378      ; je      short sndc_info_4
3379 000126CD 20DB      and     bl, bl
3380 000126CF 7546      jnz     short sndc_info_4
3381
3382 000126D1 881D[DC880100]      mov     ebx, [audio_vendor]
3383 000126D7 8B0D[D8880100]      mov     ecx, [audio_dev_id]
3384      ; mov     al, [audio_device]
3385 000126DD 3C02      cmp     al, 2 ; AC'97 (ICH)
3386 000126DF 7513      jne     short sndc_info_1
3387      ; Intel AC97 (ICH) Audio Controller (=2)
3388 000126E1 668B15[D6880100]      mov     dx, [NABMBAR]
3389 000126E8 C1E210      shl     edx, 16
3390 000126EB 668B15[D4880100]      mov     dx, [NABMBAR]
3391 000126F2 EB07      jmp     short sndc_info_2
3392      sndc_info_1:
3393      ; 05/06/2017
3394      ; Note: Intel HDA code (here) is not ready yet!
3395      ; !!! SB16 or VT8233 (VT8237R) !!!
3396 000126F4 0FB715[D6880100]      movzx   edx, word [audio_io_base]
3397      sndc_info_2:
3398 000126FB 88C4      mov     ah, al ; [audio_device]
3399 000126FD A0[D2880100]      mov     al, [audio_intr]
3400
3401      ; EAX = IRQ Number in AL
3402      ; Audio Device Number in AH
3403      ; EBX = DEV/VENDOR ID
3404      ; (DDDDDDDDDDDDDDVVVVVVVVVVVVVVVV)
3405      ; ECX = BUS/DEV/FN
3406      ; (00000000BBBBBBBDDDDFFF00000000)
3407      ; EDX = NABMBAR/NAMBAR (for AC97)
3408      ; (Low word, DX = NAMBAR address)
3409      ; EDX = Base IO Addr (DX) for SB16 & VT8233
3410
3411      ; 10/06/2017
3412 00012702 A3[548E0100]      mov     [u.r0], eax
3413 00012707 8B2D[508E0100]      mov     ebp, [u.usp]
3414 0001270D 895D10      mov     [ebp+16], ebx ; ebx
3415 00012710 895514      mov     [ebp+20], edx ; edx
3416 00012713 894D18      mov     [ebp+24], ecx ; ecx
3417
3418 00012716 C3      retn
3419
3420      sndc_info_4:
3421      ; 19/11/2023
3422 00012717 3C02      cmp     al, 2 ; Intel AC97 (ICH) Audio Controller
3423 00012719 7413      je      short sndc_info_7
3424      sndc_info_5:
3425      ; return ZERO if it is not AC97 audio controller codec
3426 0001271B 31C0      xor     eax, eax ; 0
3427 0001271D 31DB      xor     ebx, ebx ; 0
3428      sndc_info_6:
3429 0001271F A3[548E0100]      mov     [u.r0], eax
3430 00012724 8B2D[508E0100]      mov     ebp, [u.usp]
3431 0001272A 895D10      mov     [ebp+16], ebx ; ebx
3432 0001272D C3      retn
3433      sndc_info_7:
3434 0001272E E8F01F0000      call    ac97_codec_info
3435 00012733 72E6      jc      short sndc_info_5
3436
3437      ; 26/11/2023 - temporary
3438      ; and     al, 0FEh ; clear VRA support bit for test
3439
3440 00012735 EBE8      jmp     short sndc_info_6
3441
3442      sound_data:
3443      ; FUNCTION = 15
3444      ; Get Current Sound data for graphics
3445      ; 30/07/2022
3446      ; 22/06/2017
3447
3448 00012737 E878000000      call    snd_dev_check
3449      ; jc      soundc_dev_err ; Device not ready !
3450      ; 30/07/2022
3451 0001273C 728A      jc      short snd_data_dev_err
3452
3453 0001273E 80FB00      cmp     bl, 0
3454 00012741 760A      jna     short sound_data_0
3455
3456      ; Only PCM OUT buffer data is valid for now!
3457 00012743 B817000000      mov     eax, ERR_INV_PARAMETER ; 23
3458 00012748 E9F4F9FFFF      jmp     sysaudio_err
3459
3460      sound_data_0:
3461 0001274D A1[F0880100]      mov     eax, [audio_dma_buff]
3462 00012752 09C0      or      eax, eax
3463      ; jz      sound_buff_error
3464      ; 30/07/2022
3465 00012754 741A      jz      short sound_data_5
3466
3467 00012756 803D[D1880100]04      cmp     byte [audio_device], 4 ; Intel HDA
3468 0001275D 744C      je      short sound_data_4 ; temporary ! (22/06/2017)
3469
3470 0001275F 21C9      and     ecx, ecx
3471      ; jnz     short sound_data_1 ; sample transfer
3472
3473      ; Return only DMA Buffer pointer/offset...
3474      ; (If DMA Buffer has been mapped to user's
3475      ; memory space; program can get graphics
3476      ; data by using only this pointer value.)
3477
3478      ; call    get_dma_buffer_offset
3479      ; ; eax = DMA buffer offset
3480      ; ; (!not half buffer offset!)
3481      ; mov     [u.r0], eax
3482      ; retn
3483
3484      ; jz      get_dma_buffer_offset
3485      ; 30/07/2022
3486 00012761 7505      jnz     short sound_data_1
3487 00012763 E95C1F0000      jmp     get_dma_buffer_offset
3488
3489      sound_data_1:
3490      ; mov     eax, [audio_dmabuff_size]
3491      ; shr     eax, 1 ; half buffer size
3492      ; cmp     ecx, eax
3493      ; ja      short sound_buff_error
3494

```

```

3495 00012768 3B0D[F4880100] <1> cmp ecx, [audio_dmabuff_size]
3496 <1> ;ja sound_buff_error
3497 <1> ; 30/07/2022
3498 0001276E 7605 <1> jna short sound_data_6
3499 <1> sound_data_5:
3500 00012770 E9ECF9FFFF <1> jmp sound_buff_error
3501 <1> sound_data_6:
3502 00012775 89D0 <1> mov eax, edx
3503 00012777 25FF0F0000 <1> and eax, PAGE_OFF ; 4095 (0FFFh)
3504 0001277C 81F900100000 <1> cmp ecx, 4096
3505 00012782 7604 <1> jna short sound_data_2
3506 <1> ;mov ecx, 4096 ; max. 1 page
3507 <1> ; 30/07/2022
3508 00012784 31C9 <1> xor ecx, ecx
3509 00012786 B510 <1> mov ch, 16
3510 <1> ; ecx = 4096
3511 <1> sound_data_2:
3512 00012788 01C8 <1> add eax, ecx
3513 0001278A 3D00100000 <1> cmp eax, 4096
3514 0001278F 7606 <1> jna short sound_data_3
3515 00012791 6625FF0F <1> and ax, PAGE_OFF ; 4095 (0FFFh)
3516 00012795 29C1 <1> sub ecx, eax
3517 <1> ; here, ECX has been adjusted to fit
3518 <1> ; in page border.. (<= 4096, >0)
3519 <1> sound_data_3:
3520 00012797 51 <1> push ecx
3521 00012798 52 <1> push edx
3522 00012799 89D3 <1> mov ebx, edx
3523 0001279B E86434FFFF <1> call get_physical_addr
3524 000127A0 5A <1> pop edx
3525 000127A1 59 <1> pop ecx
3526 <1> ;jc sound_buff_error
3527 <1> ; 30/07/2022
3528 000127A2 72CC <1> jc short sound_data_5
3529 <1>
3530 <1> ; eax = physical address of user's buffer
3531 000127A4 89C3 <1> mov ebx, eax
3532 <1> ; ecx = byte (transfer) count
3533 <1> ;call get_current_sound_data
3534 <1> ;retn
3535 000127A6 E96E1E0000 <1> jmp get_current_sound_data
3536 <1>
3537 <1> sound_data_4:
3538 <1> ; Intel HDA code is not ready yet !
3539 <1> ; 22/06/2017
3540 000127AB 31C0 <1> xor eax, eax
3541 000127AD 48 <1> dec eax
3542 000127AE A3[548E0100] <1> mov [u.r0], eax ; 0FFFFFFFFh
3543 000127B3 C3 <1> retn
3544 <1>
3545 <1> snd_dev_check:
3546 <1> ; 10/06/2017
3547 <1> ; 05/06/2017
3548 <1> ; 24/05/2017
3549 <1> ; 22/04/2017
3550 <1> ; 21/04/2017
3551 <1> ; ... device check at first
3552 000127B4 A0[D1880100] <1> mov al, [audio_device]
3553 000127B9 3C01 <1> cmp al, 1 ; SB 16
3554 000127BB 7203 <1> jb short snd_dev_chk_retn ; error !
3555 <1> ;cmp al, 4 ; Intel HDA
3556 <1> ;ja short snd_dbchk_stc ; invalid !
3557 <1> ; 10/06/2017
3558 000127BD 3C05 <1> cmp al, 5
3559 000127BF F5 <1> cmc
3560 <1> snd_dev_chk_retn:
3561 000127C0 C3 <1> retn
3562 <1>
3563 <1> snd_buf_check:
3564 <1> ; 10/06/2017
3565 <1> ; 22/04/2017
3566 <1> ; 21/04/2017
3567 <1> ; ... buffer & (buffer) owner check at second
3568 000127C1 833D[E4880100]00 <1> cmp dword [audio_buffer], 0
3569 000127C8 760D <1> jna short snd_dbchk_stc
3570 <1> snd_user_check:
3571 000127CA A0[A58E0100] <1> mov al, [u.uno]
3572 000127CF 3A05[FD880100] <1> cmp al, [audio_user]
3573 <1> ;jne short snd_dbchk_stc
3574 <1> ;retn
3575 000127D5 74E9 <1> je short snd_dev_chk_retn
3576 <1>
3577 <1> snd_dbchk_stc:
3578 000127D7 F9 <1> stc
3579 000127D8 C3 <1> retn
3580 <1>
3581 <1> sound_update:
3582 <1> ; FUNCTION = 16
3583 <1> ; bl =
3584 <1> ; 0 = automatic (sequential) update (with flag switch!)
3585 <1> ; 1 = update dma half buffer 1 (without flag switch!)
3586 <1> ; 2 = update dma half buffer 2 (without flag switch!)
3587 <1> ; FFh = get current flag value
3588 <1> ; 0 = dma half buffer 1 (will be played next)
3589 <1> ; 1 = dma half buffer 2 (will be played next)
3590 <1>
3591 <1> ; 29/12/2024
3592 <1> ; Note: Requested half buffer is updated and
3593 <1> ; flag is set to other value for other half buffer
3594 <1>
3595 <1> ; For example:
3596 <1> ; *if BL input is 1, half buffer 1 is loaded and
3597 <1> ; flag is switched to 1 -half buff 2 is in order-
3598 <1> ; (next time half buffer 2 is updated automaticly)
3599 <1> ; *if BL input is 2, half buffer 2 is updated/loaded
3600 <1> ; and flag is switched to 0
3601 <1>
3602 <1> ; Return value for BL input = 0,1,2
3603 <1> ; Current flag value (just after switched)
3604 <1> ; -flag will have same value-
3605 <1> ; If BL input is 1, return value will be 1 (1 xor 0)
3606 <1> ; If BL input is 2, return value will be 0 (1 xor 1)
3607 <1>
3608 <1> ; 28/12/2024
3609 <1> ; 05/06/2024
3610 <1> ; 30/07/2022
3611 <1> ; 10/10/2017
3612 <1> ; ... device check at first
3613 000127D9 A0[D1880100] <1> mov al, [audio_device]
3614 000127DE 08C0 <1> or al, al ; 0 ; pc speaker or invalid
3615 <1> ;jz soundc_dev_err
3616 <1> ; 30/07/2022
3617 000127E0 7505 <1> jnz short snd_update_4
3618 000127E2 E94EF9FFFF <1> jmp soundc_dev_err

```

```

3619 <1> snd_update_4:
3620 <1> ; ... buffer & (buffer) owner check at second
3621 000127E7 833D[E4880100]00 <1> cmp dword [audio_buffer], 0
3622 <1> ;jna sound_buff_error
3623 <1> ; 30/07/2022
3624 000127EE 761C <1> jna short snd_update_6 ; jmp sound_buff_error
3625 000127F0 A0[A58E0100] <1> mov al, [u.uno]
3626 000127F5 3A05[FD880100] <1> cmp al, [audio_user]
3627 <1> ;jne sndc_owner_error
3628 <1> ; 30/07/2022
3629 000127FB 7405 <1> je short snd_update_5
3630 000127FD E985FAFFFF <1> jmp sndc_owner_error
3631 <1> snd_update_5:
3632 <1> ; Transfer ring 3 (user's) audio buffer content to dma buffer
3633 00012802 8B3D[F0880100] <1> mov edi, [audio_dma_buff] ; dma buffer (ring 0)
3634 00012808 09FF <1> or edi, edi
3635 <1> ;jz sound_buff_error
3636 <1> ; 30/07/2022
3637 0001280A 7505 <1> jnz short snd_update_7
3638 <1> snd_update_6:
3639 0001280C E950F9FFFF <1> jmp sound_buff_error
3640 <1> snd_update_7:
3641 00012811 8B35[E8880100] <1> mov esi, [audio_p_buffer] ; physical address (ring 3)
3642 <1> ; 05/06/2024
3643 <1> ;mov ecx, [audio_buff_size]
3644 <1> ;
3645 <1> ;
3646 <1> ; 04/06/2024
3647 <1> ;mov al, [audio_device]
3648 <1> ;cmp al, 1
3649 <1> ;je short snd_update_8 ; SB16
3650 <1> ;and cl, ~1 ; word alignment
3651 <1> ;cmp al, 3
3652 <1> ;je short snd_update_8 ; VIA VT8233
3653 <1> ;; al = 2 ; AC97
3654 <1> ;and cl, ~7 ; 8 byte alignment
3655 <1> ;snd_update_8:
3656 <1> ;
3657 <1> ;
3658 <1> ; 05/06/2024
3659 00012817 8B0D[F8880100] <1> mov ecx, [dma_hbuff_size] ; DMA half buffer size
3660 <1> ;
3661 <1> ;movzx eax, byte [audio_flag]
3662 0001281D A0[FC880100] <1> mov al, [audio_flag]
3663 00012822 FEC3 <1> inc bl
3664 00012824 742F <1> jz short snd_update_3 ; bl = 0FFh
3665 00012826 FECB <1> dec bl
3666 00012828 7417 <1> jz short snd_update_0 ; bl = 0
3667 <1> ;
3668 <1> ; 28/12/2024
3669 0001282A 80FB02 <1> cmp bl, 2
3670 0001282D 7609 <1> jna short snd_update_8
3671 <1> ;cmp bl, 2
3672 <1> ;je short snd_update_1 ; dma half buffer 2
3673 <1> ;jb short snd_update_2 ; dma half buffer 1
3674 <1> ; 28/12/2024
3675 <1> ;jz short snd_update_2
3676 <1> ;
3677 <1> ; invalid parameter !
3678 <1> ; 30/07/2022
3679 <1> ;mov eax, ERR_INV_PARAMETER ; 23
3680 <1> ; mov [u.r0], eax
3681 <1> ; mov [u.error], eax
3682 <1> ; jmp error
3683 <1> ;
3684 <1> snd_update_err:
3685 <1> ; 30/07/2022
3686 0001282F 29C0 <1> sub eax, eax
3687 00012831 B017 <1> mov al, ERR_INV_PARAMETER ; 23
3688 00012833 E909F9FFFF <1> jmp sysaudio_err
3689 <1> ;
3690 <1> ; 29/12/2024
3691 <1> snd_update_8:
3692 00012838 88D8 <1> mov al, bl ; 1 or 2
3693 0001283A FEC8 <1> dec al
3694 0001283C A2[FC880100] <1> mov [audio_flag], al ; 0 or 1
3695 <1> snd_update_0:
3696 00012841 8035[FC880100]01 <1> xor byte [audio_flag], 1 ; update flag !!!
3697 <1> snd_update_9:
3698 00012848 3C01 <1> cmp al, 1
3699 0001284A 7202 <1> jb short snd_update_2 ; dma half buffer 1
3700 <1> snd_update_1:
3701 <1> ; dma half buffer 2
3702 0001284C 01CF <1> add edi, ecx
3703 <1> snd_update_2:
3704 <1> ;rep movsb
3705 <1> ; 05/06/2024
3706 <1> ;shr ecx, 2
3707 <1> ;rep movsd
3708 0001284E F3A4 <1> rep movsb ; SB16, AC97, VT8233
3709 <1> ; 29/12/2024
3710 00012850 A0[FC880100] <1> mov al, [audio_flag]
3711 <1> snd_update_3:
3712 00012855 A3[548E0100] <1> mov [u.r0], eax
3713 <1> ;
3714 0001285A C3 <1> retn
3715 <1> ;
3716 <1> sound_getvol:
3717 <1> ; FUNCTION = 17
3718 <1> ; Get sound volume level
3719 <1> ; 24/05/2024
3720 <1> ; bl = component
3721 <1> ; 0 = master/playback/lineout volume
3722 <1> ; 1 = PCM out volume
3723 <1> ;
3724 <1> ; Return:
3725 <1> ; cl = left channel volume level (0 to 31)
3726 <1> ; ch = right channel volume level (0 to 31)
3727 0001285B 80FB01 <1> cmp bl, 1
3728 0001285E 7709 <1> ja short snd_gvol_0 ; temporary ; 24/05/2024
3729 00012860 7408 <1> je short snd_gvol_1
3730 00012862 668B0D[0C890100] <1> mov cx, [audio_master_volume]
3731 <1> snd_gvol_0:
3732 00012869 C3 <1> retn
3733 <1> snd_gvol_1:
3734 0001286A 668B0D[0E890100] <1> mov cx, [audio_pcmo_volume]
3735 00012871 C3 <1> retn
3736 <1> ;
3737 <1> %if 0
3738 <1> ;
3739 <1> sound_dmaclear:
3740 <1> ; FUNCTION = 18
3741 <1> ; Clear DMA buffer
3742 <1> ; (may be useful for non-VRA AC97 hardware)

```



```

3743 <1> ; b1 = cleaning byte value
3744 <1> ; 0 or 80h (not a necessary value, 0 is most proper)
3745 <1> ;
3746 <1> ; Return value
3747 <1> ; eax = DMA half buffer size (DMA buff size / 2)
3748 <1> ; If eax = 0
3749 <1> ; there is not an active DMA buffer for user)
3750 <1> ;
3751 <1> ; 28/12/2024
3752 <1> mov edi, [audio_dma_buff]
3753 <1> and edi, edi
3754 <1> jz short snd_dma_clear_ok
3755 <1> mov ecx, [audio_dmabuff_size]
3756 <1> or ecx, ecx
3757 <1> jz short snd_dma_clear_ok
3758 <1> ; security (is necessary for multitasking)
3759 <1> mov cl, [audio_user]
3760 <1> or cl, cl
3761 <1> jz short snd_dma_clear_ok
3762 <1> cmp cl, [u.uno]
3763 <1> jne short snd_dma_clear_ok
3764 <1> mov ecx, [dma_hbuff_size]
3765 <1> and ecx, ecx
3766 <1> jz short snd_dma_clear_ok
3767 <1>
3768 <1> mov [u.r0], ecx
3769 <1> shl ecx, 1 ; full buffer size in bytes
3770 <1> mov al, b1
3771 <1> rep stosb
3772 <1>
3773 <1> snd_dma_clear_ok:
3774 <1> retn
3775 <1>
3776 <1> %endif
3777 <1>
3778 <1> set_irq_callback_service:
3779 <1> ; 23/11/2023
3780 <1> ; 20/11/2023 (TRDOS 386 Kernel v2.0.7)
3781 <1> ; 30/07/2022 (TRDOS 386 Kernel v2.0.5)
3782 <1> ; 03/08/2020
3783 <1> ; 10/06/2017
3784 <1> ; 12/05/2017
3785 <1> ; 24/04/2017
3786 <1> ; 22/04/2017
3787 <1> ; caller: 'syscalbac' or 'sysaudio' or ...
3788 <1> ; 13/04/2017, 14/04/2017, 17/04/2017
3789 <1> ; 24/02/2017, 26/02/2017, 28/02/2017
3790 <1> ; 21/02/2017 - TRDOS 386 (TRDOS v2.0)
3791 <1>
3792 <1> ; Link or unlink IRQ callback service to/from user (ring 3)
3793 <1>
3794 <1> INPUT ->
3795 <1> ; If AL = 0, the caller is 'syscalbac';
3796 <1> ; otherwise, the caller is 'sysaudio' or ...
3797 <1> ; (AL = user number)
3798 <1> ;
3799 <1> ; BL = IRQ number (Hardware interrupt request number)
3800 <1> ; (0 to 15 but IRQ 0,1,2,6,8,14,15 are prohibited)
3801 <1> ; IRQ numbers 3,4,5,7,9,10,11,12,13 are valid
3802 <1> ; (numbers >15 are invalid)
3803 <1> ;
3804 <1> ; BH = 0 = Unlink IRQ (in BL) from user (ring 3) service
3805 <1> ; 1 = Link IRQ by using Signal Response Byte method
3806 <1> ; 2 = Link IRQ by using Callback service method
3807 <1> ; 3 = Link IRQ by using Auto Increment S.R.B. method
3808 <1> ; >3 = invalid
3809 <1> ; (syscallback version will return to user)
3810 <1> ;
3811 <1> ; CL = Signal Return/Response Byte value
3812 <1> ;
3813 <1> ; If BH = 3, kernel will put a counter value ; 03/08/2020
3814 <1> ; (into the S.R.B. addr)
3815 <1> ; between 0 to 255. (start value = CL+1)
3816 <1> ;
3817 <1> ; NOTE: counter value, for example: even and odd numbers
3818 <1> ; may be used for -audio- DMA buffer switch
3819 <1> ; within double buffer method, etc.
3820 <1> ;
3821 <1> ; EDX = Signal return (Response) byte address
3822 <1> ; - or -
3823 <1> ; Interrupt/Callback service/routine address
3824 <1> ;
3825 <1> ; (virtual address in user's memory space)
3826 <1>
3827 <1> OUTPUT ->
3828 <1> ; CF = 0 & EAX = 0 -> Successful setting
3829 <1> ; CF = 1 & EAX > 0 -> IRQ is prohibited or locked
3830 <1> ; by another process
3831 <1> ; eax = ERR_PERM_DENIED -> prohibited or locked
3832 <1> ; eax = ERR_INV_PARAMETER ->
3833 <1> ; invalid parameter/option or bad address
3834 <1>
3835 <1> TRDOS 386 - IRQ CALLBACK structures (parameters):
3836 <1>
3837 <1> ; [u.irqlock] : 1 word, IRQ flags (0-15) that indicates
3838 <1> ; which IRQs are locked by (that) user.
3839 <1> ; Lock and unlock (by user) will change
3840 <1> ; these flags or 'terminate process' (sysexit)
3841 <1> ; will clear these flags and unlock those IRQs.
3842 <1> ;
3843 <1> ; Bit 0 is for IRQ 0 and Bit 15 is for IRQ 15
3844 <1> ;
3845 <1> ; IRQ(x).owner : 1 byte, user, [u.uno], 0 = free (unlocked)
3846 <1> ;
3847 <1> ; IRQ(x).method : 1 byte for callback method & status
3848 <1> ; 0 = Signal Response Byte method
3849 <1> ; 1 = Callback service method
3850 <1> ; >1 = invalid for current 'syscalbac'.
3851 <1> ; or(+) 80h = IRQ is in use by system (ring 0)
3852 <1> ; function (audio etc.) or
3853 <1> ; a device driver.
3854 <1> ; (system function will ignore the lock/owner)
3855 <1> ;
3856 <1> ; IRQ(x).srb : 1 byte, Signal Return/Response byte value
3857 <1> ; (a fixed value by user or a counter value
3858 <1> ; from 0 to 255, which is increased by every
3859 <1> ; interrupt just before putting it into
3860 <1> ; the Signal Response byte address)
3861 <1> ; (This is not used in callback serv method)
3862 <1> ;
3863 <1> ; IRQ(x).addr : 1 dword
3864 <1> ; Signal Response Byte address (physical)
3865 <1> ; -or-
3866 <1> ; callback service address (virtual)

```

```

3867 <1> ;
3868 <1> ; IRQ(x).dev : 1 byte
3869 <1> ; 0 = Default device or kernel function
3870 <1> ; -or-
3871 <1> ; 1-255 = Assigned device driver number
3872 <1> ;
3873 <1> ; (x) = 3,4,5,7,9,10,11,12,13
3874 <1> ;
3875 <1> ;
3876 00012872 80FB0F <1> cmp bl, 15
3877 00012875 7727 <1> ja short scbs_2
3878 <1>
3879 00012877 80FF03 <1> cmp bh, 3
3880 0001287A 7722 <1> ja short scbs_2 ; invalid parameter
3881 <1>
3882 0001287C 0FB6FB <1> movzx edi, bl ; save IRQ number
3883 <1>
3884 <1> ; IRQ 0,1,2,6,8,14,15 are prohibited
3885 <1> ; IRQenum: ; 'trdosk9.s'
3886 <1> ; db 0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0
3887 <1>
3888 0001287F 0FB6B7[A0350100] <1> movzx esi, byte [edi+IRQenum] ; IRQ availability
3889 <1> ; enumeration/index
3890 <1> ; 20/11/2023
3891 00012886 4E <1> dec esi
3892 <1> ; dec si
3893 00012887 780F <1> js short scbs_1 ; 0 -> 0FFFFh
3894 <1>
3895 <1> ; ESI = IRQ callback parameters index number (0 to 8)
3896 <1>
3897 00012889 08FF <1> or bh, bh
3898 0001288B 7417 <1> jz short scbs_4 ; unlink the IRQ (in BL)
3899 <1>
3900 0001288D FECF <1> dec bh
3901 <1> ; bh = method (0 = signal response byte, 1 = callback)
3902 <1> ; (2 = auto increment of signal response byte)
3903 <1>
3904 0001288F 80BE[80880100]00 <1> cmp byte [esi+IRQ.owner], 0 ; locked ?
3905 00012896 7635 <1> jna short scbs_6 ; no... OK...
3906 <1>
3907 <1> scbs_1:
3908 <1> ; permission denied (prohibited IRQ)
3909 <1> ; mov eax, ERR_PERM_DENIED
3910 <1> ; 30/07/2022
3911 00012898 29C0 <1> sub eax, eax
3912 0001289A B00B <1> mov al, ERR_PERM_DENIED
3913 0001289C F9 <1> stc
3914 0001289D C3 <1> retn
3915 <1> scbs_2:
3916 <1> ; stc
3917 <1> scbs_3:
3918 <1> ; mov eax, ERR_INV_PARAMETER
3919 <1> ; 30/07/2022
3920 0001289E 29C0 <1> sub eax, eax
3921 000128A0 B017 <1> mov al, ERR_INV_PARAMETER
3922 000128A2 F9 <1> stc
3923 000128A3 C3 <1> retn
3924 <1>
3925 <1> scbs_4: ; unlink the requested IRQ (if it belongs to current user)
3926 <1> ; 10/06/2017
3927 <1> ; 22/04/2017
3928 <1> ; 14/04/2017
3929 <1> ; If AL = 0 -> The caller is 'syscalbac'
3930 000128A4 8AA6[80880100] <1> mov ah, [esi+IRQ.owner]
3931 000128AA 3A25[A58E0100] <1> cmp ah, [u.uno]
3932 000128B0 75E6 <1> jne short scbs_1
3933 <1>
3934 000128B2 FE0D[CE8E0100] <1> dec byte [u.irqc] ; decrease IRQ count (in use)
3935 <1>
3936 <1> ; sub ah, ah
3937 <1> ; mov [esi+IRQ.owner], ah ; 0 ; free !!!
3938 <1> ; and byte [esi+IRQ.method], 80h
3939 <1> ; mov [esi+IRQ.srb], ah ; 0
3940 <1> ; mov [esi+IRQ.dev], ah ; 0
3941 <1> ; mov dword [esi+IRQ.addr], 0
3942 <1> ; mov dword [u.r0], 0
3943 <1>
3944 <1> ; mov byte [esi+IRQ.owner], 0
3945 <1>
3946 <1> ; 22/04/2017
3947 000128B8 29C0 <1> sub eax, eax
3948 000128BA 8886[80880100] <1> mov [esi+IRQ.owner], al ; 0
3949 <1> ; 10/06/2017
3950 000128C0 8686[92880100] <1> xchg al, [esi+IRQ.method]
3951 000128C6 2480 <1> and al, 80h
3952 000128C8 745D <1> jz short scbs_12
3953 <1> ; Audio device must be disabled -later- ! ([IRQ.medhod] = 80h)
3954 <1>
3955 <1> ; cmp byte [esi+IRQ.method], 80h ; device drv or kernel extension ?
3956 <1> ; jb short scbs_12 ; bh = 0 reset to default IRQ handler
3957 <1> ;
3958 <1> ; and al, al
3959 <1> ; jz short scbs_5 ; the caller is 'syscalbac'
3960 <1> ; The caller is 'sysaudio' or ...
3961 000128CA 30C0 <1> xor al, al
3962 <1> ; mov [esi+IRQ.method], al ; 0 ; reset kernel extension flag
3963 <1> ; scbs_5:
3964 <1> ; sub ah, ah
3965 <1> ; mov [u.r0], eax ; 0
3966 000128CC C3 <1> retn
3967 <1>
3968 <1> scbs_6:
3969 <1> ; 14/04/2017
3970 000128CD 20C0 <1> and al, al
3971 000128CF 7405 <1> jz short scbs_7 ; the caller is 'syscalbac'
3972 <1> ; AL = user number ([u.uno] or [audio.user] or ...)
3973 <1> ; The caller is 'sysaudio' or ...
3974 <1> ;
3975 <1> ; bh = method (0 = signal response byte, 1 = callback)
3976 <1> ; (2 = auto increment of signal response byte)
3977 <1> ;
3978 000128D1 80CF80 <1> or bh, 80h ; kernel extension flag !
3979 000128D4 EB0A <1> jmp short scbs_8
3980 <1> scbs_7:
3981 000128D6 8A86[92880100] <1> mov al, [esi+IRQ.method] ; >= 80h = kernel is using this IRQ
3982 000128DC 2480 <1> and al, 80h ; use only bit 7 (kernel function flag)
3983 000128DE 08C7 <1> or bh, al ; method
3984 <1> ; 0 = signal response byte, 1 = callback
3985 <1> ; 2 = auto increment of s.r.b.
3986 <1> scbs_8:
3987 000128E0 A0[A58E0100] <1> mov al, [u.uno] ; user (process) number (1 to 16)
3988 000128E5 8886[80880100] <1> mov [esi+IRQ.owner], al ; lock the IRQ for user
3989 000128EB 88BE[92880100] <1> mov [esi+IRQ.method], bh
3990 <1>

```

```

3991      <1> ; test bh, 1
3992      <1> ; jnz short scbs_9 ; Callback method, CX will not be used
3993      <1> ;
3994      <1> ; test bh, 2 ; use auto increment (counter) method
3995      <1> ; jz short scbs_10 ; (count can be used for buffer switch)
3996      <1> ; scbs_9:
3997      <1> ; xor ecx, ecx ; 0
3998      <1> ;
3999      <1> scbs_10:
4000      <1> ; mov [esi+IRQ.method], bh
4001      <1> ; mov [esi+IRQ.srb], cl
4002      <1> ; mov byte [esi+IRQ.dev], 0 ; device number is always 0
4003      <1> ; ; for this system call
4004      <1> ;
4005      <1> ; test bh, 1
4006      <1> ; and bh, 1 ; 17/04/2017
4007      <1> ; jnz short scbs_11 ; callback method, use virtual address
4008      <1> ;
4009      <1> ; push ebx ; IRQ number (in BL)
4010      <1> ; mov ebx, edx
4011      <1> ; ; ebx = virtual address
4012      <1> ; ; [u.pgdir] = page directory's physical address
4013      <1> ; inc byte [no_page_swap] ; 1
4014      <1> ; ; Do not add this page to swap queue
4015      <1> ; ; and remove it from swap queue if it is
4016      <1> ; ; on the queue.
4017      <1> ; call get_physical_addr
4018      <1> ; pop ebx
4019      <1> ; jc short scbs_3 ; invalid address !
4020      <1> ; ; eax = physical address of the virtual address in user's space
4021      <1> ; mov edx, eax
4022      <1> ;
4023      <1> scbs_11:
4024      <1> ; shl si, 2 ; byte (index) to dword (offset)
4025      <1> ; ; 23/11/2023
4026      <1> ; shl esi, 2
4027      <1> ; mov [esi+IRQ.addr], edx
4028      <1> ;
4029      <1> ; inc byte [u.irqc] ; increase IRQ (in use) count
4030      <1> ;
4031      <1> ; inc bh ; 17/04/2017
4032      <1> ; ; bh > 0 -> set to requested IRQ handler (IRQ_u_list)
4033      <1> ;
4034      <1> scbs_12:
4035      <1> ; mov al, bl ; IRQ number
4036      <1> ; mov ah, bh ; 0 = reset, >0 = set
4037      <1> ; call set_hardware_int_vector
4038      <1> ;
4039      <1> ; xor eax, eax
4040      <1> ; mov [u.r0], eax ; 0
4041      <1> ;
4042      <1> ; retn ; return with success (cf=0, eax=0)
4043      <1> ;
4044      <1> sysdma: ; DMA FUNCTIONS
4045      <1> ; ; 08/08/2022
4046      <1> ; ; 30/07/2022 - TRDOS 386 kernel v2.0.5
4047      <1> ; ; 02/09/2017
4048      <1> ; ; 28/08/2017
4049      <1> ; ; 20/08/2017 - TRDOS 386 (TRDOS v2.0)
4050      <1> ;
4051      <1> ; Inputs:
4052      <1> ; ; BH = 0 -> Allocate DMA buffer
4053      <1> ; ; BL = 0 -> Use the system's default DMA
4054      <1> ; ; (SB16) Buffer
4055      <1> ; ; Buffer Size (max.) = 65536 bytes
4056      <1> ; ; BL > 0 -> Allocate (a new) DMA buffer
4057      <1> ; ; ECX = DMA Buffer Size in bytes (<=128KB)
4058      <1> ; ; EDX = Virtual Address of DMA buffer
4059      <1> ; ;
4060      <1> ; ; BH = 1 -> Initialize (Start) DMA service
4061      <1> ; ; BL, bit 0 to 3 = Channel Number (0 to 7)
4062      <1> ; ; BL, bit 7 = Auto Initialized Mode
4063      <1> ; ; (If bit 7 is set)
4064      <1> ; ; bit 6 = Record (read) mode (0= playback)
4065      <1> ; ; ECX = byte count (0 = use dma buffer size)
4066      <1> ; ; EDX = physical buffer address
4067      <1> ; ; (0 = use dma buffer -start- address)
4068      <1> ; ;
4069      <1> ; ; BH = 2 -> Get Current DMA Buffer Offset
4070      <1> ; ; BL = DMA channel number
4071      <1> ; ;
4072      <1> ; ; BH = 3 -> Get Current DMA count down value
4073      <1> ; ; BL = DMA channel number (0 to 7)
4074      <1> ; ;
4075      <1> ; ; BH = 4 -> Get Current DMA channel (in progress)
4076      <1> ; ;
4077      <1> ; ; BH = 5 -> Get System's Default DMA Buffer Address
4078      <1> ; ;
4079      <1> ; ; BH = 6 -> Get Current DMA Buffer Address
4080      <1> ; ;
4081      <1> ; ; BH = 7 -> Stop DMA service
4082      <1> ;
4083      <1> ; Outputs:
4084      <1> ; ;
4085      <1> ; ; For BH = 0 ; Allocate DMA buffer
4086      <1> ; ; EAX = Physical address of DMA buffer
4087      <1> ; ; ECX = Allocated buffer size in bytes
4088      <1> ; ; - page count * 4096 -
4089      <1> ; ; (may be bigger than requested)
4090      <1> ; ; If BL input > 0,
4091      <1> ; ; 'sysalloc:' system call will be used with
4092      <1> ; ; EBX (for 'sysalloc') = EDX (for 'sysdma')
4093      <1> ; ; ECX is same, byte count (buffer size)
4094      <1> ; ; EDX = 1024*1024*16 ; 16 MB upper limit
4095      <1> ; ; If BL input = 0,
4096      <1> ; ; Default DMA buffer (SB16 buffer) will be
4097      <1> ; ; checked and if it is free, it's address
4098      <1> ; ; will be returned in EAX and it's size
4099      <1> ; ; will be returned in ECX (as 65536)
4100      <1> ; ;
4101      <1> ; ; If CF = 1, error code is in EAX
4102      <1> ; ; EAX = -1 ; DMA buffer allocation error!
4103      <1> ; ; EAX = 11 ; 'Permission Denied' error !
4104      <1> ; ;
4105      <1> ; ; Note: 'sysalloc' error return method
4106      <1> ; ; will be applied if BL input > 0 !
4107      <1> ; ;
4108      <1> ; ; For BH = 1 ; Initialize (Start) DMA
4109      <1> ; ; EAX = 0 (Successful)
4110      <1> ; ; If CF = 1, error code is in EAX
4111      <1> ; ;
4112      <1> ; ; For BH = 2 ; Get Current DMA Buffer Offset
4113      <1> ; ; EAX = DMA Buffer Offset (in bytes)
4114      <1> ; ;
4115      <1> ; ; ;
4116      <1> ; ; AX = DMA buffer offset
4117      <1> ; ; EAX bits 16 to 23 = Page register value

```

```

4115 <1> ; For BH = 3 ; Get Current DMA count down value
4116 <1> ; EAX = Count down value (remain bytes)
4117 <1> ;
4118 <1> ; For BH = 4 ; Get Current DMA channel (in progress)
4119 <1> ; EAX = DMA channel number (0 to 7)
4120 <1> ; AH = 0 if the owner is the caller process
4121 <1> ; AH > 0 if the dma channel is in use by
4122 <1> ; another user/process
4123 <1> ; EAX = -1 (0FFFFFFFh)
4124 <1> ; if DMA service is not in use
4125 <1> ; (stopped or not initialized/started)
4126 <1> ;
4127 <1> ; For BH = 5 ; Get System's Default DMA Buff Addr
4128 <1> ; EAX = Default DMA Buffer Address (Physical)
4129 <1> ; = offset 'sb16_dma_buffer:'
4130 <1> ; ECX = Buffer size
4131 <1> ; = 65536
4132 <1> ;
4133 <1> ; For BH = 6 ; Get Current DMA Buffer Address
4134 <1> ; EAX = Current DMA buffer address (Physical)
4135 <1> ; ECX = Current DMA buffer size (setting value)
4136 <1> ; Note: These values are for current dma channel
4137 <1> ; settings for the user/process
4138 <1> ; ** For now (for current TRDOS 386 version)
4139 <1> ; only one user/process can use only one
4140 <1> ; dma channel & one dma buffer at same time
4141 <1> ; (no multi tasking on DMA service) !!! **
4142 <1> ; (Once, current DMA user must stop it's own DMA
4143 <1> ; DMA service, than another user/program
4144 <1> ; can use DMA service with same dma channel
4145 <1> ; or with another DMA channel.)
4146 <1> ;
4147 <1> ; For BH = 7 ; Stop DMA service (for current user
4148 <1> ; and current DMA channel)
4149 <1> ; EAX = 0 ; successful
4150 <1> ; CF = 1 & EAX > 0 (= -1) -> Error
4151 <1> ;
4152 00012933 80FF07 <1> cmp bh, 7
4153 00012936 7612 <1> jna short sysdma_0
4154 <1> ;
4155 <1> sysdma_err:
4156 00012938 31C0 <1> xor eax, eax
4157 0001293A 48 <1> dec eax ; -1
4158 <1> sysdma_perm_err:
4159 0001293B A3[548E0100] <1> mov [u.r0], eax
4160 00012940 A3[C08E0100] <1> mov [u.error], eax ; DMA service error !
4161 00012945 E914A1FFFF <1> jmp error
4162 <1> ;
4163 <1> sysdma_0:
4164 0001294A 08FF <1> or bh, bh
4165 0001294C 7537 <1> jnz short sysdma_1 ; 30/07/2022
4166 <1> ;
4167 0001294E 20DB <1> and bl, bl
4168 00012950 7416 <1> jz short sysdma_01
4169 <1> ;
4170 <1> ; redirect system call to 'sysalloc'
4171 00012952 89D3 <1> mov ebx, edx ; virtual address of DMA buffer
4172 <1> ; ecx = Buffer size in bytes
4173 <1> ; DMA buffer address <= 16MB upper limit
4174 00012954 BA00000001 <1> mov edx, 1024*1024*16 ; 16MB limit for DMA buff
4175 <1> ;
4176 00012959 C705[188D0100]FFFF- <1> mov dword [dma_addr], 0FFFFFFFh ; -1
4177 00012961 FFFF <1> ;
4178 <1> jmp sysalloc
4179 <1> ;
4180 <1> sysdma_01:
4181 00012968 B800000500 <1> mov eax, sb16_dma_buffer
4182 <1> ;
4183 0001296D 803D[D1880100]01 <1> cmp byte [audio_device], 1
4184 00012974 724A <1> jb short sysdma_03
4185 <1> ;
4186 00012976 3B05[F0880100] <1> cmp eax, [audio_dma_buff]
4187 0001297C 7527 <1> jne short sysdma_02
4188 <1> ;
4189 <1> sysdma_0_err:
4190 0001297E B80B000000 <1> mov eax, ERR_PERM_DENIED
4191 00012983 EBB6 <1> jmp short sysdma_perm_err
4192 <1> ;
4193 <1> ; 30/07/2022
4194 <1> sysdma_1:
4195 00012985 80FF01 <1> cmp bh, 1
4196 <1> ; ja sysdma_5
4197 <1> ; 30/07/2022
4198 00012988 7605 <1> jna short sysdma_10
4199 0001298A E90C010000 <1> jmp sysdma_5
4200 <1> ;
4201 <1> sysdma_10:
4202 0001298F F6C340 <1> test bl, 40h ; record (read) mode -BL, bit 6-
4203 <1> ; jnz sysdma_err ; not ready yet!
4204 <1> ; 30/07/2022
4205 00012992 757C <1> jnz short sysdma_06 ; jmp sysdma_err
4206 <1> ;
4207 00012994 A1[188D0100] <1> mov eax, [dma_addr] ; physical address of dma buffer
4208 00012999 21C0 <1> and eax, eax
4209 <1> ; jz sysdma_err
4210 <1> ; 30/07/2022
4211 0001299B 7473 <1> jz short sysdma_06 ; jmp sysdma_err
4212 <1> ;
4213 0001299D 09D2 <1> or edx, edx
4214 0001299F 7574 <1> jnz short sysdma_11
4215 <1> ;
4216 000129A1 89C2 <1> mov edx, eax
4217 000129A3 EB74 <1> jmp short sysdma_12
4218 <1> ;
4219 <1> sysdma_02:
4220 <1> ; Only one user is permitted for audio/dma functions
4221 <1> ;
4222 000129A5 833D[F0880100]00 <1> cmp dword [audio_dma_buff], 0
4223 000129AC 7612 <1> jna short sysdma_03
4224 <1> ;
4225 000129AE 8A1D[FD880100] <1> mov bl, [audio_user]
4226 000129B4 08DB <1> or bl, bl
4227 000129B6 7408 <1> jz short sysdma_03
4228 <1> ;
4229 000129B8 3A1D[A58E0100] <1> cmp bl, [u.uno]
4230 000129BE 75BE <1> jne short sysdma_0_err
4231 <1> ;
4232 <1> sysdma_03:
4233 000129C0 8A1D[158D0100] <1> mov bl, [dma_user]
4234 000129C6 20DB <1> and bl, bl
4235 000129C8 750E <1> jnz short sysdma_04
4236 <1> ;
4237 000129CA 8A1D[A58E0100] <1> mov bl, [u.uno]

```

```

4238 000129D0 881D[158D0100] <1> mov [dma_user], b1
4239 <1>
4240 000129D6 EB15 <1> jmp short sysdma_05
4241 <1>
4242 <1> sysdma_04:
4243 000129D8 8B35[188D0100] <1> mov esi, [dma_addr]
4244 000129DE 21F6 <1> and esi, esi
4245 000129E0 740B <1> jz short sysdma_05
4246 <1>
4247 000129E2 46 <1> inc esi ; -1 -> 0
4248 000129E3 7408 <1> jz short sysdma_05
4249 <1>
4250 000129E5 3A1D[A58E0100] <1> cmp b1, [u.uno]
4251 000129EB 7591 <1> jne short sysdma_0_err
4252 <1>
4253 <1> sysdma_05:
4254 <1> ; edx = virtual address (user's buffer address)
4255 <1> ;
4256 000129ED 81F900000100 <1> cmp ecx, 65536 ; byte count (buffer size)
4257 <1> ;ja sysdma_err
4258 <1> ; 30/07/2022
4259 000129F3 771B <1> ja short sysdma_06 ; jmp sysdma_err
4260 <1> ;
4261 000129F5 81C1FF0F0000 <1> add ecx, PAGE_SIZE-1 ; 4095
4262 000129FB 6681E100F0 <1> and cx, ~PAGE_OFF ; not 4095
4263 <1> ;cmp ecx, 65536
4264 <1> ;ja sysdma_err
4265 00012A00 51 <1> push ecx ; buffer size (allocated pages * 4096)
4266 00012A01 50 <1> push eax ; offset sb16_dma_buffer
4267 00012A02 89D3 <1> mov ebx, edx
4268 00012A04 C1E90C <1> shr ecx, 12 ; byte count to page count
4269 <1> ; eax = physical address of (audio) dma buffer
4270 <1> ; ebx = virtual address of (audio) dma buffer (user's pgdir)
4271 <1> ; ecx = page count (>0)
4272 00012A07 E8B735FFFF <1> call direct_memory_access
4273 00012A0C 58 <1> pop eax
4274 00012A0D 59 <1> pop ecx
4275 <1> ;jc sysdma_err
4276 <1> ; 30/07/2022
4277 00012A0E 7315 <1> jnc short sysdma_07
4278 <1> sysdma_06:
4279 00012A10 E923FFFFFF <1> jmp sysdma_err
4280 <1>
4281 <1> sysdma_11:
4282 00012A15 39C2 <1> cmp edx, eax
4283 <1> ;jb sysdma_err
4284 <1> ; 30/07/2022
4285 00012A17 72F7 <1> jb short sysdma_06 ; jmp sysdma_err
4286 <1> sysdma_12:
4287 00012A19 21C9 <1> and ecx, ecx
4288 00012A1B 7515 <1> jnz short sysdma_13
4289 <1>
4290 00012A1D 8B0D[1C8D0100] <1> mov ecx, [dma_size]
4291 00012A23 EB15 <1> jmp short sysdma_14
4292 <1>
4293 <1> sysdma_07:
4294 00012A25 A3[188D0100] <1> mov [dma_addr], eax
4295 00012A2A 890D[1C8D0100] <1> mov [dma_size], ecx ; dma buffer size (in bytes)
4296 <1>
4297 <1> ;mov [u.r0], eax ; DMA Buffer Address (Physical)
4298 <1>
4299 <1> ;mov ebp, [u.usp] ; ebp points to user's registers
4300 <1> ;mov [ebp+24], ecx ; return to user with ecx value
4301 <1>
4302 <1> ;jmp sysret
4303 <1>
4304 <1> ; 28/08/2017
4305 00012A30 EB7A <1> jmp sysdma_51
4306 <1>
4307 <1> sysdma_13:
4308 00012A32 3B0D[1C8D0100] <1> cmp ecx, [dma_size]
4309 <1> ;ja sysdma_err
4310 <1> ; 30/07/2022
4311 00012A38 77D6 <1> ja short sysdma_06 ; jmp sysdma_err
4312 <1> sysdma_14:
4313 00012A3A 89C6 <1> mov esi, eax
4314 00012A3C 0335[1C8D0100] <1> add esi, [dma_size]
4315 <1>
4316 00012A42 89D0 <1> mov eax, edx
4317 00012A44 01C8 <1> add eax, ecx
4318 <1> ;jc sysdma_err ; 02/09/2017
4319 <1> ; 30/07/2022
4320 00012A46 72C8 <1> jc short sysdma_06 ; jmp sysdma_err
4321 <1>
4322 00012A48 39F0 <1> cmp eax, esi
4323 <1> ;ja sysdma_err
4324 <1> ; 30/07/2022
4325 00012A4A 77C4 <1> ja short sysdma_06 ; jmp sysdma_err
4326 <1>
4327 00012A4C 8B3D[F0880100] <1> mov edi, [audio_dma_buff]
4328 00012A52 8B35[188D0100] <1> mov esi, [dma_addr]
4329 <1>
4330 00012A58 09FF <1> or edi, edi
4331 00012A5A 7425 <1> jz short sysdma_16
4332 <1>
4333 00012A5C 803D[D1880100]01 <1> cmp byte [audio_device], 1
4334 00012A63 7209 <1> jb short sysdma_15
4335 <1>
4336 <1> ; Sound Blaster 16
4337 00012A65 39FE <1> cmp esi, edi
4338 <1> ;je sysdma_0_err ; permission denied !
4339 <1> ; 30/07/2022
4340 00012A67 7505 <1> jne short sysdma_15
4341 00012A69 E910FFFFFF <1> jmp sysdma_0_err
4342 <1> sysdma_15:
4343 00012A6E C605[178D0100]48 <1> mov byte [dma_mode], 48h ; single mode playback
4344 <1>
4345 00012A75 F6C380 <1> test b1, 80h ; DMA mode - BL, bit 7, auto init -
4346 00012A78 7407 <1> jz short sysdma_16
4347 <1> ; Auto initialized playback (write) mode
4348 00012A7A 8005[178D0100]10 <1> add byte [dma_mode], 10h ; = 58h
4349 <1> sysdma_16:
4350 00012A81 80E307 <1> and b1, 07h
4351 00012A84 881D[168D0100] <1> mov [dma_channel], b1
4352 00012A8A 8915[208D0100] <1> mov [dma_start], edx
4353 00012A90 890D[248D0100] <1> mov [dma_count], ecx
4354 <1>
4355 <1> ; 28/08/2017
4356 <1> ;call dma_init
4357 <1> ;jmp sysret
4358 00012A96 E945010000 <1> jmp dma_init
4359 <1>
4360 <1> sysdma_5:
4361 00012A9B 80FF05 <1> cmp bh, 5

```

```

4362 00012A9E 726D      <1>      jb      short sysdma_3
4363 00012AA0 7759      <1>      ja      short sysdma_6
4364                                <1>
4365                                <1>      ; Get the system's default dma buffer addr and size
4366 00012AA2 B800000500 <1>      mov     eax, sb16_dma_buffer
4367 00012AA7 B900000100 <1>      mov     ecx, 65536 ; Buffer size in bytes
4368                                <1>
4369                                <1> sysdma_51:
4370                                <1>      ; 0 = there is not a dma buffer (in use or available)
4371 00012AAC A3[548E0100] <1>      mov     [u.r0], eax
4372                                <1>
4373 00012AB1 8B2D[508E0100] <1>      mov     ebp, [u.usp] ; ebp points to user's registers
4374 00012AB7 894D18      <1>      mov     [ebp+24], ecx ; return to user with ecx value
4375                                <1>
4376 00012ABA E9BF9FFFFF <1>      jmp      sysret
4377                                <1>
4378                                <1> sysdma_2:
4379                                <1>      ; Get current dma buffer offset (& page)
4380                                <1>      ; 28/08/2017
4381 00012ABF 0FB635[168D0100] <1>      movzx   esi, byte [dma_channel]
4382 00012AC6 0FB696[D8350100] <1>      movzx   edx, byte [dma_flip+esi]
4383 00012ACD EE          <1>      out     dx, al ; flip-flop clear
4384 00012ACE 8A96[B0350100] <1>      mov     dl, [dma_adr+esi]
4385 00012AD4 EC          <1>      in     al, dx ; get dma position
4386 00012AD5 0FB6D8      <1>      movzx   ebx, al
4387 00012AD8 EC          <1>      in     al, dx
4388 00012AD9 88C7      <1>      mov     bh, al
4389                                <1>
4390 00012ADB 6683FE04      <1>      cmp     si, 4 ; channel number ?
4391 00012ADF 7202      <1>      jb      short sysdma_21 ; 8 bit dma channel
4392                                <1>
4393 00012AE1 D1E3      <1>      shl     ebx, 1 ; word offset to byte offset
4394                                <1>
4395                                <1> sysdma_21:
4396 00012AE3 891D[548E0100] <1>      mov     [u.r0], ebx
4397                                <1>
4398 00012AE9 8A96[C0350100] <1>      mov     dl, [dma_page+esi]
4399 00012AEF EC          <1>      in     al, dx ; get dma page
4400                                <1>
4401                                <1>      ; add [u.ro+2], al
4402 00012AF0 0805[568E0100] <1>      or      [u.r0+2], al
4403                                <1>
4404 00012AF6 E9839FFFFF <1>      jmp      sysret
4405                                <1>
4406                                <1> sysdma_6:
4407 00012AFB 80FF06      <1>      cmp     bh, 6
4408 00012AFE 775C      <1>      ja      short sysdma_7
4409                                <1>
4410                                <1>      ; 28/08/2017
4411                                <1>      ; Get current DMA buffer addr and size
4412 00012B00 A1[188D0100] <1>      mov     eax, [dma_addr] ; dma buffer address
4413 00012B05 8B0D[1C8D0100] <1>      mov     ecx, [dma_size] ; dma buffer size (in bytes)
4414                                <1>
4415 00012B0B EB9F      <1>      jmp      short sysdma_51
4416                                <1>
4417                                <1> sysdma_3:
4418 00012B0D 80FF03      <1>      cmp     bh, 3
4419 00012B10 72AD      <1>      jb      short sysdma_2
4420 00012B12 772F      <1>      ja      short sysdma_4
4421                                <1>
4422                                <1>      ; Get current dma count down value (remain bytes)
4423                                <1>      ; 28/08/2017
4424 00012B14 0FB635[168D0100] <1>      movzx   esi, byte [dma_channel]
4425 00012B1B 0FB696[D8350100] <1>      movzx   edx, byte [dma_flip+esi]
4426 00012B22 EE          <1>      out     dx, al ; flip-flop clear
4427 00012B23 8A96[B8350100] <1>      mov     dl, [dma_cnt+esi] ; dma count register addr
4428 00012B29 EC          <1>      in     al, dx
4429 00012B2A 0FB6D8      <1>      movzx   ebx, al
4430 00012B2D EC          <1>      in     al, dx
4431 00012B2E 88C7      <1>      mov     bh, al
4432                                <1>
4433 00012B30 6683FE04      <1>      cmp     si, 4 ; channel number ?
4434 00012B34 7202      <1>      jb      short sysdma_31 ; 8 bit dma channel
4435                                <1>
4436 00012B36 D1E3      <1>      shl     ebx, 1 ; word count to byte count
4437                                <1>
4438                                <1> sysdma_31:
4439 00012B38 891D[548E0100] <1>      mov     [u.r0], ebx
4440                                <1>
4441 00012B3E E93B9FFFFF <1>      jmp      sysret
4442                                <1>
4443                                <1> sysdma_4:
4444                                <1>      ; Get current DMA channel number
4445                                <1>      ; 28/08/2017
4446 00012B43 8A25[158D0100] <1>      mov     ah, [dma_user]
4447 00012B49 20E4      <1>      and     ah, ah
4448 00012B4B 7539      <1>      jnz     short sysdma_42
4449                                <1>
4450                                <1> sysdma_41:
4451                                <1>      ; Not a valid dma channel (in use)
4452 00012B4D C705[548E0100]FFFF- <1>      mov     dword [u.r0], -1 ; 0FFFFFFFh
4453 00012B55 FFFF      <1>
4454 00012B57 E9229FFFFF <1>      jmp      sysret
4455                                <1>
4456                                <1> sysdma_7:
4457 00012B5C A0[A58E0100] <1>      ; DMA service STOP
4458 00012B61 3A05[158D0100] <1>      mov     al, [u.uno]
4459 00012B67 7543      <1>      cmp     al, [dma_user]
4460                                <1>      jne     short sysdma_72
4461 00012B69 28C0      <1>      sub     al, al ; 0
4462                                <1>
4463 00012B6B A2[158D0100] <1>      mov     [dma_user], al ; clear user
4464                                <1>
4465 00012B70 8605[178D0100] <1>      xchg    al, [dma_mode]
4466 00012B76 20C0      <1>      and     al, al
4467                                <1>      ; jz short sysdma_err
4468 00012B78 754E      <1>      jnz     short sysdma_73
4469                                <1>
4470                                <1> sysdma_71:
4471 00012B7A 31C0      <1>      xor     eax, eax
4472 00012B7C A3[548E0100] <1>      mov     [u.r0], eax ; 0
4473 00012B81 E9F89EFFFF <1>      jmp      sysret
4474                                <1>
4475                                <1> sysdma_42:
4476 00012B86 8B35[188D0100] <1>      mov     esi, [dma_addr]
4477 00012B8C 21F6      <1>      and     esi, esi
4478 00012B8E 74BD      <1>      jz      short sysdma_41
4479                                <1>
4480 00012B90 46          <1>      inc     esi ; -1 -> 0
4481 00012B91 74BA      <1>      jz      short sysdma_41
4482                                <1>
4483 00012B93 A0[168D0100] <1>      mov     al, [dma_channel]
4484                                <1>

```

```

4485 00012B98 3A25[A58E0100] <1> cmp ah, [u.uno]
4486 00012B9E 7502 <1> jne short sysdma_43
4487 <1>
4488 00012BA0 30E4 <1> xor ah, ah ; DMA channel in use by current user
4489 <1>
4490 <1> sysdma_43:
4491 00012BA2 A3[548E0100] <1> mov [u.r0], eax ; AL = dma channel number
4492 <1> ; AH > 0 if the the channel
4493 <1> ; in use by another user/process
4494 00012BA7 E9D29EFFFF <1> jmp sysret
4495 <1>
4496 <1> sysdma_72:
4497 <1> ; 28/08/2017
4498 00012BAC 803D[158D0100]00 <1> cmp byte [dma_user], 0
4499 00012BB3 76C5 <1> jna short sysdma_71 ; Nothing to do !
4500 <1>
4501 00012BB5 833D[188D0100]00 <1> cmp dword [dma_addr], 0
4502 <1> ;ja sysdma_0_err
4503 <1> ; 30/07/2022
4504 00012BBC 7605 <1> jna short sysdma_74
4505 00012BBE E9BBFDFFFF <1> jmp sysdma_0_err
4506 <1>
4507 <1> sysdma_74:
4508 00012BC3 A2[158D0100] <1> mov [dma_user], al ; reset to current user
4509 <1>
4510 <1> sysdma_73:
4511 <1> ; 28/08/2017
4512 00012BC8 0FB635[168D0100] <1> movzx esi, byte [dma_channel]
4513 00012BCF 0FB696[C8350100] <1> movzx edx, byte [dma_mask+esi]
4514 00012BD6 A0[168D0100] <1> mov al, [dma_channel]
4515 00012BDB 0C04 <1> or al, 4
4516 00012BDD EE <1> out dx, al
4517 <1>
4518 00012BDE EB9A <1> jmp short sysdma_71
4519 <1>
4520 <1> dma_init:
4521 <1> ; 30/07/2022 (TRDOS 386 kernel v2.0.5)
4522 <1> ; 28/08/2017
4523 <1> ; 20/08/2017
4524 <1> ; DMA initialization
4525 <1> ; 14/08/2017
4526 <1> ; 03/08/2017, 06/08/2017, 08/08/2017
4527 <1> ; 02/07/2017, 13/07/2017, 16/07/2017, 30/07/2017
4528 <1> ; (Derived from 'DMA_INIT' procedure in SB16MOD.ASM)
4529 <1> ; Modified for TRDOS 386 DMA buffer allocation & initialization !
4530 <1>
4531 00012BE0 8B1D[208D0100] <1> mov ebx, [dma_start]
4532 00012BE6 8B0D[248D0100] <1> mov ecx, [dma_count]
4533 <1>
4534 00012BEC 0FB635[168D0100] <1> movzx esi, byte [dma_channel]
4535 <1>
4536 00012BF3 6683FE04 <1> cmp si, 4
4537 00012BF7 7204 <1> jb short gdmi1
4538 <1> ; 08/08/2017
4539 <1> ;shr cx, 1 ; word count
4540 <1> ; 30/07/2022
4541 00012BF9 D1E9 <1> shr ecx, 1
4542 00012BFB D1EB <1> shr ebx, 1 ; convert byte offset to word offset
4543 <1>
4544 <1> gdmi1:
4545 <1> ;mov [dma_poff], bx ; 08/08/2017
4546 <1> ;dec cx ; dma size = block size - 1
4547 <1> ; 30/07/2022
4548 <1> dec ecx
4549 <1>
4549 00012BFE 0FB696[C8350100] <1> movzx edx, byte [dma_mask+esi] ; 30/07/2017
4550 00012C05 A0[168D0100] <1> mov al, [dma_channel]
4551 00012C0A 0C04 <1> or al, 4
4552 00012C0C EE <1> out dx, al ; dma channel mask
4553 <1>
4554 00012C0D 30C0 <1> xor al, al ; 0 ; any value ! 08/08/2017
4555 00012C0F 8A96[D8350100] <1> mov dl, [dma_flip+esi]
4556 00012C15 EE <1> out dx, al ; flip-flop clear
4557 <1>
4558 00012C16 8A96[D0350100] <1> mov dl, [dma_mod+esi]
4559 00012C1C A0[168D0100] <1> mov al, [dma_channel] ; 13/07/2017
4560 00012C21 2403 <1> and al, 3
4561 <1> ; 08/08/2017
4562 00012C23 0A05[178D0100] <1> or al, [dma_mode] ; 58h ; dma mode for SB16
4563 00012C29 EE <1> out dx, al
4564 <1>
4565 00012C2A 8A96[B0350100] <1> mov dl, [dma_adr+esi]
4566 00012C30 88D8 <1> mov al, bl
4567 00012C32 EE <1> out dx, al ; offset low
4568 <1>
4569 00012C33 88F8 <1> mov al, bh
4570 00012C35 EE <1> out dx, al ; offset high
4571 <1>
4572 00012C36 8A96[B8350100] <1> mov dl, [dma_cnt+esi]
4573 00012C3C 88C8 <1> mov al, cl
4574 00012C3E EE <1> out dx, al ; size low
4575 <1>
4576 00012C3F 88E8 <1> mov al, ch
4577 00012C41 EE <1> out dx, al ; size high
4578 <1>
4579 00012C42 8A96[C0350100] <1> mov dl, [dma_page+esi]
4580 <1> ; 14/08/2017
4581 00012C48 6683FE04 <1> cmp si, 4
4582 00012C4C 7305 <1> jnb short gdmi2
4583 00012C4E C1EB10 <1> shr ebx, 16
4584 00012C51 EB06 <1> jmp short gdmi3
4585 <1>
4586 <1> gdmi2:
4587 00012C53 C1EB0F <1> shr ebx, 15 ; complete 16 bit shift
4588 00012C56 80E3FE <1> and bl, 0FEh ; clear bit 0 (not necessary)
4589 <1>
4590 00012C59 88D8 <1> gdmi3:
4591 00012C5B EE <1> mov al, bl
4592 <1> out dx, al ; page
4593 <1>
4593 00012C5C 8A96[C8350100] <1> mov dl, [dma_mask+esi]
4594 00012C62 A0[168D0100] <1> mov al, [dma_channel] ; 13/07/2017
4595 00012C67 2403 <1> and al, 3
4596 00012C69 EE <1> out dx, al ; dma channel unmask
4597 <1>
4598 <1> ;retn
4599 <1> ; 28/08/2017
4600 00012C6A E90F9EFFFF <1> jmp sysret
4601 <1>
4602 <1> sysstdio: ; STDIN/STDOUT/STDERR functions
4603 <1> ; 18/09/2024
4604 <1> ; 07/09/2024
4605 <1> ; (STDAUX/STDPRN functions, pre-definions)
4606 <1> ; (!these functions are not ready!)
4607 <1> ; 24/08/2024
4608 <1> ; 23/08/2024

```

```

4609      <1> ; 20/08/2024 - TRDOS 386 kernel v2.0.9
4610      <1>
4611      <1>
4612      <1> Inputs:
4613      <1> BL = 0 -> read a character on stdin (wait)
4614      <1> BL = 1 -> read a character on stdin (no wait)
4615      <1> BL = 2 -> write a character onto stdout (redirection)
4616      <1> BL = 3 -> write a character onto stderr (no redirection)
4617      <1> BL = 4 -> redirect stdin to file (if cl > 0)
4618      <1> BL = 5 -> redirect stdout to file (if cl > 0)
4619      <1> BL = 6 -> read character (ascii and scancode) on stdin
4620      <1> ; -no redirection, wait-
4621      <1> BL = 7 -> read character (ascii and scancode) on stdin
4622      <1> ; -no redirection, no wait-
4623      <1> BL = 8 -> write character and color onto stdout
4624      <1> ; -no redirection-
4625      <1> BL = 9 -> ungetchar (put back the ascii code in u.getc)
4626      <1>
4627      <1> For BL=2,3,8,9
4628      <1> ; CL = character (ascii) code
4629      <1> For BL=8
4630      <1> ; CH = color (Attribute) -CGA-
4631      <1> For BL=4,5
4632      <1> ; CL = file descriptor number + 1
4633      <1> ; (File must be open and that number is 'u.fp' index)
4634      <1>
4635      <1> 07/09/2024 (these subfunctions will not be handled
4636      <1> ; by kernel v2.0.9 for now)
4637      <1> -I am writing them here for C compiler compatibility,
4638      <1> ; for now. Ref: SCC STDIO.H-
4639      <1> BL = 10 -> get STDAUX status
4640      <1> BL = 11 -> get/select STDAUX (COM) port
4641      <1> ; (clears redirection)
4642      <1> BL = 12 -> redirect STDAUX
4643      <1> BL = 13 -> STDAUX IOCTL (control functions)
4644      <1> BL = 14 -> read byte/character from STDAUX
4645      <1> BL = 15 -> write byte/character to STDAUX
4646      <1> BL = 16 -> get STDPN status (LPT printer only)
4647      <1> BL = 17 -> redirect STDPN
4648      <1> BL = 18 -> STDPN IOCTL (init/control functions)
4649      <1> BL = 19 -> write byte/character to STDPN
4650      <1>
4651      <1> For BL = 11
4652      <1> ; If CL = 0 -> get STDAUX (COM) port number
4653      <1> ; CL = 1 -> COM1
4654      <1> ; CL = 2 -> COM2
4655      <1> ; CL > 4 -> invalid
4656      <1> For BL = 12
4657      <1> ; If ECX = 0 -> clear redirection
4658      <1> ; If CH & 7Fh = 0 -> redirect INPUT
4659      <1> ; CH & 7Fh > 0 -> redirect OUTPUT
4660      <1> ; If CH & 80h = 80h
4661      <1> ; CL = file handle (descriptor index)
4662      <1> ; If CH & 80h = 0
4663      <1> ; CL = console (pseudo) tty number,
4664      <1> ; 1 to 8 for tty0 to tty7.
4665      <1> For BL = 13
4666      <1> ; If CL = 0 -> get control byte
4667      <1> ; If CL > 1 -> set control byte (in CL)
4668      <1> ; bit 0 = 1
4669      <1> ; bit 1 = data bits (set=8, clear=7)
4670      <1> ; bit 2 = stop bit (set=2, clear=1)
4671      <1> ; bit 3 = parity bit (set=yes, clear=no)
4672      <1> ; bit 4 = parity bit (set=even, clear=odd)
4673      <1> ; bit 5-6-7 = data rate
4674      <1> ; 000 = 9600
4675      <1> ; 001 = 19200
4676      <1> ; 010 = 38400
4677      <1> ; 011 = 57600
4678      <1> ; 100 = 115200
4679      <1> For BL = 15, 19
4680      <1> ; CL = byte/character to r/w
4681      <1> For BL = 17
4682      <1> ; If ECX = 0 -> clear redirection
4683      <1> ; If CH & 80h = 80h
4684      <1> ; CL = file handle (descriptor index)
4685      <1> ; If CH & 80h = 0
4686      <1> ; CL = console (pseudo) tty number,
4687      <1> ; 1 to 8 for tty0 to tty7.
4688      <1> For BL = 18
4689      <1> ; If CL = 0 -> initialize printer (command=0)
4690      <1> ; CL > 0 -> configuration/command byte
4691      <1> ; (reserved, not used)
4692      <1>
4693      <1> Outputs:
4694      <1>
4695      <1> For BL=0,1,2,3,6,7,8,9
4696      <1> ; AL = character (ascii) code
4697      <1> For BL=6,7
4698      <1> ; AH = scan code
4699      <1> For BL=4,5
4700      <1> ; AL = file descriptor (CL input - 1)
4701      <1>
4702      <1> If CF=1
4703      <1> ; AL (EAX) = error code
4704      <1> ; If EAX = 0 -> EOF ; 18/09/2024
4705      <1>
4706      <1> 07/09/2024
4707      <1> For BL = 10 to 19
4708      <1> ; If CF = 1, EAX/AL = error code
4709      <1> .. If CF = 0 ...
4710      <1> For BL = 10 & 16
4711      <1> ; AL (EAX) = status
4712      <1> For BL = 11
4713      <1> ; EAX/AL = COM (serial) port (1 to 4)
4714      <1> For BL = 12, 15, 17, 18, 19
4715      <1> ; EAX/AL = 0
4716      <1> For BL = 13
4717      <1> ; If CL input = 0
4718      <1> ; EAX/AL = current ctrl/cfg byte
4719      <1> ; If CL input > 0
4720      <1> ; EAX/AL = ctrl/cfg byte
4721      <1> For BL = 14
4722      <1> ; EAX/AL = byte/character
4723      <1>
4724      <1> cmp bl, (end_of_stdiofuncs-stdiofuncs)>>2
4725      <1> jb short sysstdio_0
4726      <1>
4727      <1> ; invalid parameter (invalid sub function number)
4728      <1> mov eax, ERR_INV_PARAMETER
4729      <1> sysstdio_err:
4730      <1> mov [u.r0], eax
4731      <1> mov [u.error], eax
4732      <1> jmp error

```



```

4733 <1>
4734 <1> sysstdio_0:
4735 00012C88 31C0 <1> xor     eax, eax ; 0
4736 00012C8A A3[C08E0100] <1> mov     [u.error], eax ; clear previous error code
4737 00012C8F A3[548E0100] <1> mov     [u.r0], eax ; clear previous return code
4738 <1>
4739 00012C94 0FB6F3 <1> movzx   esi, bl
4740 00012C97 C1E602 <1> shl     esi, 2
4741 00012C9A FFA6[A02C0100] <1> jmp     dword [esi+stdiofuncs]
4742 <1>
4743 <1> stdiofuncs:
4744 00012CA0 [2A2D0100] <1> dd      readstdinw
4745 00012CA4 [2A2D0100] <1> dd      readstdinw
4746 00012CA8 [412E0100] <1> dd      writestdout
4747 00012CAC [F82D0100] <1> dd      writestderr
4748 00012CB0 [C82C0100] <1> dd      readstdin
4749 00012CB4 [062D0100] <1> dd      readstdout
4750 <1> stdinacsc:
4751 00012CB8 [162E0100] <1> dd      readstdin2w
4752 00012CBC [162E0100] <1> dd      readstdin2w
4753 <1> stdoutcc:
4754 00012CC0 [4C2E0100] <1> dd      writestdoutcc
4755 00012CC4 [632E0100] <1> dd      ungetchar
4756 <1> end_of_stdiofuncs:
4757 <1>
4758 <1> readstdin:
4759 <1> ; CL = file handle (descriptor/index) + 1
4760 00012CC8 08C9 <1> or      cl, cl
4761 00012CCA 7508 <1> jnz     short readstdin_0
4762 00012CCC 880D[BC8E0100] <1> mov     [u.stdin], cl ; 0 ; reset STDIN to keyboard
4763 00012CD2 EB24 <1> jmp     short readstdin_3
4764 <1> readstdin_0:
4765 00012CD4 88C8 <1> mov     al, cl
4766 00012CD6 E8B3D5FFFF <1> call    getf
4767 <1> ; eax = first cluster
4768 <1> ; ebx = system (not user) open file index
4769 00012CDB 80BB[04840100]01 <1> cmp     byte [ebx+OF_MODE], 1 ; open for read
4770 00012CE2 7407 <1> je      short readstdin_1
4771 <1> redir_acc_denied:
4772 00012CE4 B805000000 <1> mov     eax, ERR_ACCESS_DENIED
4773 00012CE9 EB8E <1> jmp     short sysstdio_err
4774 <1> readstdin_1:
4775 00012CEB 880D[BC8E0100] <1> mov     [u.stdin], cl
4776 <1> readstdout_2:
4777 00012CF1 49 <1> dec     ecx
4778 00012CF2 880D[548E0100] <1> mov     [u.r0], cl ; return file descriptor
4779 <1> readstdin_3:
4780 <1> readstdout_3:
4781 00012CF8 66C705[BE8E0100]00- <1> mov     word [u.ungetc], 0
4782 00012D00 00 <1>
4783 <1> ; reset u.ungetc and u.getc
4784 00012D01 E9789DFFFF <1> jmp     sysret
4785 <1>
4786 <1> readstdout:
4787 <1> ; CL = file handle (descriptor/index) + 1
4788 00012D06 08C9 <1> or      cl, cl
4789 00012D08 7508 <1> jnz     short readstdout_0
4790 00012D0A 880D[BD8E0100] <1> mov     [u.stdout], cl ; 0 ; reset STDOUT to keyboard
4791 00012D10 EBE6 <1> jmp     short readstdout_3
4792 <1> readstdout_0:
4793 00012D12 88C8 <1> mov     al, cl
4794 00012D14 E875D5FFFF <1> call    getf
4795 <1> ; eax = first cluster
4796 <1> ; ebx = system (not user) open file index
4797 00012D19 80BB[04840100]02 <1> cmp     byte [ebx+OF_MODE], 2 ; open for write
4798 00012D20 75C2 <1> jne     short redir_acc_denied
4799 <1> readstdout_1:
4800 00012D22 880D[BD8E0100] <1> mov     [u.stdout], cl
4801 00012D28 EBC7 <1> jmp     short readstdout_2
4802 <1>
4803 <1> readstdinw:
4804 <1> readstdinw:
4805 <1> ; read a character on stdin (wait)
4806 00012D2A 3805[BE8E0100] <1> sub     eax, eax
4807 00012D30 761E <1> cmp     byte [u.ungetc], al ; 0
4808 <1> jna     short readstdinw_0
4809 00012D32 A2[BE8E0100] <1> ; read the character in u.getc buffer (put by ungetchar)
4810 00012D37 8605[BF8E0100] <1> mov     [u.ungetc], al ; 0
4811 <1> xchg    al, [u.getc]
4812 <1> readstdinw_retn:
4813 00012D42 E9379DFFFF <1> mov     [u.r0], al
4814 <1> jmp     sysret
4815 <1>
4816 <1> readstdinw_3:
4817 00012D47 B410 <1> ; 23/08/2024
4818 00012D49 E8CEE1FEFF <1> mov     ah, 10h
4819 00012D4E EBED <1> call    int16h
4820 <1> jmp     short readstdinw_retn
4821 <1>
4822 <1> readstdinw_0:
4823 00012D50 A0[BC8E0100] <1> mov     al, [u.stdin]
4824 00012D55 08C0 <1> or      al, al
4825 00012D57 745F <1> jz      short readstdinw_1
4826 <1>
4827 <1> readstdinw_1:
4828 00012D59 48 <1> ; 24/08/2024
4829 <1> dec     eax
4830 <1>
4831 <1> ; file
4832 00012D5A 89C3 <1> mov     ebx, eax ; File handle (descriptor/index)
4833 <1> ; 18/09/2024
4834 00012D5C B9[782E0100] <1> mov     ecx, charbuf ; buffer address
4835 00012D61 BA01000000 <1> mov     edx, 1 ; 1 character only
4836 <1>
4837 <1> ; 18/09/2024
4838 <1> ;;;
4839 00012D66 E825D5FFFF <1> call    getf1
4840 00012D6B 722B <1> jc      short redirect_fno_err
4841 <1>
4842 <1> readstdinw_@:
4843 <1> ; eax = first cluster
4844 <1> ; ebx = file descriptor (system, open files)
4845 00012D6D E882A4FFFF <1> call    rw1
4846 00012D72 7229 <1> jc      short redirection_err
4847 <1>
4848 00012D74 FE05[BA8E0100] <1> inc     byte [u.kcall]
4849 <1> ;mov     byte [u.kcall], 1 ; >0 means system buffer
4850 <1> ; (buff addr is not virtual)
4851 00012D7A E8F7D8FFFF <1> call    readi
4852 <1>
4853 <1> redirected_rw_OK:
4854 00012D7F 31C0 <1> xor     eax, eax
4855 00012D81 3905[808E0100] <1> cmp     [u.nread], eax ; 0

```

```

4856 00012D87 7623      <1>      jna      short readstdinf_EOF
4857                                <1>
4858 00012D89 A0[7B2E0100] <1>      mov      al, [charbuf]
4859                                <1> ;redirected_w_not_OK:
4860 00012D8E A3[548E0100] <1>      mov      [u.r0], eax
4861 00012D93 E9E69CFFFF <1>      jmp      sysret
4862                                <1>
4863                                <1> redirect_fno_err:
4864 00012D98 B80A000000 <1>      mov      eax, ERR_FILE_NOT_OPEN ; file not open !
4865                                <1> redirection_err:
4866 00012D9D A3[C08E0100] <1>      mov      [u.error], eax
4867                                <1> readstdinf_EOF_@:
4868 00012DA2 A3[548E0100] <1>      mov      [u.r0], eax ; error code ; 0 = EOF
4869 00012DA7 E9B29CFFFF <1>      jmp      error
4870                                <1>
4871                                <1> readstdinf_EOF:
4872                                <1>      ; 'end of file !' error
4873 00012DAC C705[C08E0100]1000- <1>      mov      dword [u.error], ERR_FILE_EOF ; 16
4874 00012DB4 0000 <1>
4875 00012DB6 EBEA <1>      jmp      short readstdinf_EOF_@
4876                                <1>      ;;;
4877                                <1> readstdinw_1:
4878                                <1>      ; 18/09/2024
4879 00012DB8 20DB <1>      and      bl, bl
4880 00012DBA 748B <1>      jz      short readstdinw_3 ; wait (int16h, 10h)
4881                                <1>      ; no wait (int16h, 11h)
4882 00012DBC B411 <1>      mov      ah, 11h ; EXTENDED ASCII STATUS
4883 00012DBE E859E1FEFF <1>      call     int16h
4884                                <1>      ; ah = scan code, al = ascii code
4885                                <1>      ;jnz     short readstdinw_retn
4886                                <1>      ; 23/08/2024
4887 00012DC3 7582 <1>      jnz     short readstdinw_3
4888                                <1> readstdinw_2:
4889                                <1>
4890                                <1>      ; 23/08/2024
4891                                <1>      ; ; if zf=1 at here
4892                                <1>      ; ; it means 'no code available' for function 11h
4893                                <1>      ; and      bl, bl ; 0 ?
4894                                <1>      ; jz      short short readstdinw_retn ; function 10h
4895                                <1>      ; ; function 11h
4896                                <1>      ; ; [u.r0] = 0 = eax return
4897                                <1> readstdin2w@_retn:
4898 00012DC5 E9B49CFFFF <1>      jmp      sysret
4899                                <1>
4900                                <1> writestdoutf:
4901                                <1>      ; 24/08/2024
4902 00012DCA 48 <1>      dec      eax
4903                                <1>
4904                                <1>      ; file
4905 00012DCB 89C3 <1>      mov      ebx, eax ; File handle (descriptor/index)
4906                                <1>      ; 18/09/2024
4907 00012DCD 880D[7B2E0100] <1>      mov      byte [charbuf], cl
4908 00012DD3 B9[7B2E0100] <1>      mov      ecx, charbuf
4909 00012DD8 BA01000000 <1>      mov      edx, 1 ; 1 character only
4910                                <1>
4911                                <1>      ; 18/09/2024
4912                                <1>      ;;;
4913 00012DDD E8AED4FFFF <1>      call     getf1
4914 00012DE2 72B4 <1>      jc      short redirect_fno_err
4915                                <1>
4916                                <1> writestdoutf_@:
4917                                <1>      ; eax = first cluster
4918                                <1>      ; ebx = file descriptor (system, open files)
4919                                <1>
4920 00012DE4 E80BA4FFFF <1>      call     rw1
4921 00012DE9 72B2 <1>      jc      short redirection_err
4922                                <1>
4923 00012DEB FE05[BA8E0100] <1>      inc      byte [u.kcall]
4924                                <1>      ;mov     byte [u.kcall], 1 ; >0 means system buffer
4925                                <1>      ; (buff addr is not virtual)
4926 00012DF1 E859DFFFFF <1>      call     writei
4927                                <1>
4928                                <1>      ;xor     eax, eax
4929                                <1>      ;cmp     [u.nread], eax ; 0
4930                                <1>      ;jna     short redirected_w_not_OK
4931 00012DF6 EB87 <1>      jmp      short redirected_rw_OK
4932                                <1>      ;;;
4933                                <1>
4934                                <1> writestderr:
4935                                <1>      ; skip redirection
4936                                <1>      ; STDERR = STDOUT (as file redirection disabled)
4937 00012DF8 88C8 <1>      mov      al, cl ; character
4938 00012DFA A2[548E0100] <1>      mov      [u.r0], al
4939 00012DFF B40E <1>      mov      ah, 0Eh ; write a character (as tty write)
4940 00012E01 BB07000000 <1>      mov      ebx, 07h ; video page 0 (and color/attrib 07h)
4941                                <1>      ; 23/08/2024
4942 00012E06 8A3D[CE760100] <1>      mov      bh, [ptty] ; ACTIVE_PAGE
4943 00012E0C E813E9FEFF <1>      call     _int10h
4944 00012E11 E9689CFFFF <1>      jmp      sysret
4945                                <1>
4946                                <1> readstdin2w:
4947 00012E16 66C705[BE8E0100]00- <1>      mov      word [u.ungetc], 0
4948 00012E1E 00 <1>
4949 00012E1F B410 <1>      mov      ah, 10h ; Keyboard, EXTENDED READ
4950                                <1>      ;cmp     bl, 6
4951 00012E21 80FB06 <1>      cmp      bl, (stdinacsc-stdiofuncs)>>2
4952 00012E24 740B <1>      je      short readstdin2w_1 ; wait (int16h, 10h)
4953                                <1>      ; no wait (int16h, 11h)
4954 00012E26 FEC4 <1>      inc      ah ; function 11h ; EXTENDED ASCII STATUS
4955                                <1>
4956                                <1>      ; 24/08/2024
4957 00012E28 E8EFE0FEFF <1>      call     int16h
4958                                <1>      ; ah = scan code, al = ascii code
4959                                <1>      ;jnz     short readstdin2w_retn
4960                                <1>      ; 23/08/2024
4961 00012E2D 7496 <1>      jz      short readstdin2w@_retn
4962                                <1> ;readstdin2w_1:
4963                                <1>
4964                                <1>      ; 23/08/2024
4965                                <1>      ; ; if zf=1 at here
4966                                <1>      ; ; it means 'no code available' for function 11h
4967                                <1>      ; ; cmp     bl, 6
4968                                <1>      ; ; cmp     bl, (stdinacsc-stdiofuncs)>>2
4969                                <1>      ; ; je      short readstdin2w_retn ; function 10h
4970                                <1>      ; ; function 11h
4971                                <1>      ; ; [u.r0] = 0 = eax return
4972                                <1>      ; ; jmp     sysret
4973                                <1>      ; jne     short readstdin2w@_retn
4974                                <1>
4975                                <1>      ; 23/08/2024
4976 00012E2F B410 <1>      mov      ah, 10h
4977                                <1> readstdin2w_1:
4978                                <1>      ; 18/09/2024

```

```

4978 00012E31 E8E6E0FEFF <1> call int16h
4979 <1>
4980 <1> readstdin2w_retn:
4981 00012E36 66A3[548E0100] <1> mov [u.r0], ax
4982 00012E3C E93D9CFFFF <1> jmp sysret
4983 <1>
4984 <1> writestdout:
4985 <1> ; 18/09/2024
4986 00012E41 A0[BD8E0100] <1> mov al, [u.stdout]
4987 00012E46 20C0 <1> and al, al
4988 00012E48 7580 <1> jnz short writestdoutf ; 18/09/2024
4989 <1>
4990 <1> writestdout_1:
4991 <1> ;mov byte [ccolor], 07h ; default color (CGA)
4992 <1> ; (black background, light gray character)
4993 <1> ; 18/09/2024
4994 00012E4A B507 <1> mov ch, 07h
4995 <1> writestdoutcc:
4996 00012E4C BE[548E0100] <1> mov esi, u.r0 ; buffer
4997 <1> ; 18/09/2024
4998 00012E51 880E <1> mov [esi], cl ; character
4999 <1> ;cmp bl, 8 ; write char and color
5000 <1> ;cmp bl, (stdoutcc-stdiofuncs)>>2
5001 <1> ;jne short writestdout_2
5002 00012E53 882D[CF760100] <1> mov byte [ccolor], ch ; color/attribute (CGA)
5003 <1> writestdout_2:
5004 00012E59 E8A9D6FFFF <1> call print_cmsg
5005 <1> ; 18/09/2024
5006 <1> ;mov byte [ccolor], 07h ; set default color again
5007 <1>
5008 <1> ; [u.r0] = written character (AL)
5009 00012E5E E91B9CFFFF <1> jmp sysret
5010 <1>
5011 <1> ungetchar:
5012 <1> ; put a character back in u.getc STDIN buffer
5013 00012E63 C605[BE8E0100]01 <1> mov byte [u.ungetc], 1
5014 00012E6A 880D[BF8E0100] <1> mov [u.getc], cl
5015 00012E70 880D[548E0100] <1> mov [u.r0], cl
5016 00012E76 E9039CFFFF <1> jmp sysret
5017 <1>
5018 <1> ; 18/09/2024
5019 00012E7B 00 <1> charbuf: db 0
5020 <1>
5021 <1> otty:
5022 <1> sret:
5023 <1> ocvt:
5024 <1> ctty:
5025 <1> cret:
5026 <1> ccvt:
5027 <1> rtty:
5028 <1> wtty:
5029 <1> rmem:
5030 <1> wmem:
5031 <1> rfd:
5032 <1> rhd:
5033 <1> wfd:
5034 <1> whd:
5035 <1> rlpt:
5036 <1> wlpt:
5037 <1> rcvt:
5038 <1> xmtt:
5039 00012E7C C3 <1> retn
3440 <1> %include 'trdosk9.s' ; 04/01/2016
1 <1> ; *****
2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.10) - INITIALIZED DATA : trdosk9.s
3 <1> ;
4 <1> ; -----
5 <1> ; Last Update: 10/01/2026 (Previous: 29/12/2024 - Kernel v2.0.9)
6 <1> ; -----
7 <1> ; Beginning: 04/01/2016
8 <1> ; -----
9 <1> ; Assembler: NASM version 2.15 (trdos386.s)
10 <1> ; -----
11 <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
12 <1> ; TRDOS2.ASM (09/11/2011)
13 <1> ; *****
14 <1> ; DRV_INIT.ASM [26/09/2009] Last Update: 07/08/2011
15 <1> ; MAINPROG.ASM [17/01/2004] Last Update: 09/11/2011
16 <1> ; CMD_INTR.ASM [29/01/2005] Last Update: 09/11/2011
17 <1> ; FILE.ASM [29/10/2009] Last Update: 09/10/2011
18 <1>
19 <1> ; 12/02/2016
20 <1> Last_DOS_DiskNo:
21 00012E7D 01 <1> db 1 ; A: = 0 & B: = 1
22 <1>
23 <1> Restore_CDIR:
24 00012E7E FF <1> db 0FFh ; Initial value -> any number except 0
25 <1>
26 <1> msg_CRLF_temp:
27 00012E7F 070D0A00 <1> db 07h, 0Dh, 0Ah, 0
28 <1>
29 <1> Magic_Bytes:
30 00012E83 04 <1> db 4
31 00012E84 01 <1> db 1
32 <1> mainprog_Version:
33 00012E85 07 <1> db 7
34 00012E86 5B5452444F535D204D- <1> db "[TRDOS] Main Program v2.0.10 (10/01/2026)"
34 00012E8F 61696E2050726F6772- <1>
34 00012E98 616D2076322E302E31- <1>
34 00012EA1 30202831302F30312F- <1>
34 00012EAA 3230323629 <1>
35 00012EAF 0D0A <1> db 0Dh, 0Ah
36 00012EB1 286329204572646F67- <1> db "(c) Erdogan Tan 2005-2026"
36 00012EBA 616E2054616E203230- <1>
36 00012EC3 30352D32303236 <1>
37 00012ECA 0D0A00 <1> db 0Dh, 0Ah, 0
38 <1>
39 <1> MainProgCfgFile: ; 14/04/2016
40 00012ECD 4D41494E50524F472E- <1> db "MAINPROG.CFG", 0
40 00012ED6 43464700 <1>
41 <1>
42 <1> TRDOSPromptLabel:
43 00012EDA 5452444F53 <1> db "TRDOS"
44 00012EDF 00 <1> db 0
45 00012EE0 00<rep 5h> <1> times 5 db 0
46 00012EE5 00 <1> db 0
47 <1>
48 <1> ; INTERNAL COMMANDS
49 <1> Command_List:
50 00012EE6 44495200 <1> Cmd_Dir: db "DIR", 0
51 00012EEA 434400 <1> Cmd_Cd: db "CD", 0
52 00012EED 433A00 <1> Cmd_Drive: db "C:", 0
53 00012EF0 56455200 <1> Cmd_Ver: db "VER", 0
54 00012EF4 4558495400 <1> Cmd_Exit: db "EXIT", 0

```

```

55 00012EF9 50524F4D505400 <1> Cmd_Prompt: db "PROMPT", 0
56 00012F00 564F4C554D4500 <1> Cmd_Volume: db "VOLUME", 0
57 00012F07 4C4F4E474E414D4500 <1> Cmd_LongName: db "LONGNAME", 0
58 00012F10 4441544500 <1> Cmd_Date: db "DATE", 0
59 00012F15 54494D4500 <1> Cmd_Time: db "TIME", 0
60 00012F1A 52554E00 <1> Cmd_Run: db "RUN", 0
61 00012F1E 53455400 <1> Cmd_Set: db "SET", 0
62 00012F22 434C5300 <1> Cmd_Cls: db "CLS", 0
63 00012F26 53484F5700 <1> Cmd_Show: db "SHOW", 0
64 00012F2B 44454C00 <1> Cmd_Del: db "DEL", 0
65 00012F2F 41545452494200 <1> Cmd_Attrib: db "ATTRIB", 0
66 00012F36 52454E414D4500 <1> Cmd_Rename: db "RENAME", 0
67 00012F3D 524D44495200 <1> Cmd_Rmdir: db "RMDIR", 0
68 00012F43 4D4B44495200 <1> Cmd_Mkdir: db "MKDIR", 0
69 00012F49 434F505900 <1> Cmd_Copy: db "COPY", 0
70 00012F4E 4D4F564500 <1> Cmd_Move: db "MOVE", 0
71 00012F53 5041544800 <1> Cmd_Path: db "PATH", 0
72 00012F58 4D454D00 <1> Cmd_Mem: db "MEM", 0
73 00012F5C 00 <1> db 0
74 00012F5D 46494E4400 <1> Cmd_Find: db "FIND", 0
75 00012F62 4543484F00 <1> Cmd_Echo: db "ECHO", 0
76 00012F67 2A00 <1> Cmd_Remark: db "*", 0
77 00012F69 3F00 <1> Cmd_Help: db "?", 0
78 00012F6B 44455649434500 <1> Cmd_Device: db "DEVICE", 0
79 00012F72 4445564C49535400 <1> Cmd_DevList: db "DEVLIST", 0
80 00012F7A 434844495200 <1> Cmd_Chdir: db "CHDIR", 0
81 00012F80 4245455000 <1> Cmd_Beep: db "BEEP", 0
82 <1>
83 00012F85 00 <1> db 0
84 <1>
85 <1> ; 15/02/2016 (FILE.ASM, 09/10/2011)
86 <1> invalid_fname_chars:
87 00012F86 222728292A2B2C2F <1> db 22h, 27h, 28h, 29h, 2Ah, 2Bh, 2Ch, 2Fh
88 00012F8E 3A3B3C3D3E3F40 <1> db 3Ah, 3Bh, 3Ch, 3Dh, 3Eh, 3Fh, 40h
89 00012F95 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 00012F9A 456E746572206E6577- <1> db 'Enter new date (dd-mm-yy): '
94 00012FA3 206461746520286464- <1>
94 00012FAC 2D6D6D2D7979293A20 <1>
95 00012FB5 00 <1> db 0
96 <1> Msg_Show_Date:
97 00012FB6 43757272656E742064- <1> db 'Current date is '
97 00012FBF 61746520697320 <1>
98 00012FC6 30 <1> Day: db '0'
99 00012FC7 30 <1> db '0'
100 00012FC8 2F <1> db '/'
101 00012FC9 30 <1> Month: db '0'
102 00012FCA 30 <1> db '0'
103 00012FCB 2F <1> db '/'
104 00012FCC 30 <1> Century: db '0'
105 00012FCD 30 <1> db '0'
106 00012FCE 30 <1> Year: db '0'
107 00012FCF 30 <1> db '0'
108 00012FD0 0D0A00 <1> db 0Dh, 0Ah, 0
109 <1>
110 <1> Msg_Enter_Time:
111 00012FD3 456E746572206E6577- <1> db 'Enter new time: '
111 00012FDC 2074696D653A20 <1>
112 00012FE3 00 <1> db 0
113 <1> Msg_Show_Time:
114 00012FE4 43757272656E742074- <1> db 'Current time is '
114 00012FED 696D6520697320 <1>
115 00012FF4 30 <1> Hour: db '0'
116 00012FF5 30 <1> db '0'
117 00012FF6 3A <1> db ':'
118 00012FF7 30 <1> Minute: db '0'
119 00012FF8 30 <1> db '0'
120 00012FF9 3A <1> db ':'
121 00012FFA 30 <1> Second: db '0'
122 00012FFB 30 <1> db '0'
123 00012FFC 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 00012FFF 206B696C6F62797465- <1> db " kilobytes", 0Dh, 0Ah, 0
129 00013008 730D0A00 <1>
130 <1> VolSize_Bytes:
131 0001300C 2062797465730D0A00 <1> db " bytes", 0Dh, 0Ah, 0
132 <1> Volume_in_drive:
133 00013015 0D0A <1> db 0Dh, 0Ah
134 <1> Vol_FS_Name:
135 00013017 54522046533120 <1> db "TR FS1 "
136 0001301E 566F6C756D6520696E- <1> db "Volume in drive "
136 00013027 20647269766520 <1>
137 0001302E 30 <1> Vol_Drv_Name: db 30h
138 0001302F 3A <1> db ":"
139 00013030 20697320 <1> db " is "
140 00013034 0D0A00 <1> db 0Dh, 0Ah, 0
141 <1> Dir_Drive_Str:
142 00013037 54522D444F53204472- <1> db "TR-DOS Drive "
142 00013040 69766520 <1>
143 <1> Dir_Drive_Name:
144 00013044 303A <1> db "0:"
145 00013046 0D0A <1> db 0Dh, 0Ah
146 <1> Vol_Str_Header:
147 00013048 566F6C756D65204E61- <1> db "Volume Name: "
147 00013051 6D653A20 <1>
148 <1> Vol_Name:
149 00013055 00<rep 40h> <1> times 64 db 0
150 00013095 00 <1> db 0
151 <1> Vol_Serial_Header:
152 00013096 0D0A <1> db 0Dh, 0Ah
153 00013098 566F6C756D65205365- <1> db "Volume Serial No: "
153 000130A1 7269616C204E6F3A20 <1>
154 <1> Vol_Serial1:
155 000130AA 30303030 <1> db "0000"
156 000130AE 2D <1> db "-"
157 <1> Vol_Serial2:
158 000130AF 30303030 <1> db "0000"
159 000130B3 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 000130B6 0D0A <1> db 0Dh, 0Ah
165 000130B8 566F6C756D65205369- <1> db "Volume Size : ", 0
165 000130C1 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 000130C7 467265652053706163- <1> db "Free Space : ", 0
173 000130D0 6520203A2000 <1>
174 <1> ;Vol_Free_Sectors_Str:
175 <1> ; db "00000000000"
176 <1> ;Vol_Free_Sectors_Str_End:
177 <1> ; db 0
178 <1>
179 <1> Dir_Str_Header:
180 000130D6 4469726563746F7279- <1> db "Directory: "
180 000130DF 3A20 <1>
181 000130E1 2F <1> Dir_Str_Root: db "/"
182 000130E2 00<rep 40h> <1> Dir_Str: times 64 db 0
183 00013122 00000000 <1> dd 0
184 00013126 00 <1> db 0
185 <1>
186 <1> Msg_Bad_Command:
187 00013127 42616420636F6D6D61- <1> db "Bad command or file name!"
187 00013130 6E64206F722066696C- <1>
187 00013139 65206E616D6521 <1>
188 00013140 0D0A00 <1> db 0Dh, 0Ah, 0
189 <1>
190 <1> msgl_drv_not_ready:
191 00013143 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 00013146 4472697665206E6F74- <1> db "Drive not ready or read error!"
196 0001314F 207265616479206F72- <1>
196 00013158 207265616420657272- <1>
196 00013161 6F7221 <1>
197 00013164 0D0A00 <1> db 0Dh, 0Ah, 0
198 <1>
199 <1> Msg_Not_Ready_Write_Err:
200 00013167 4472697665206E6F74- <1> db "Drive not ready or write error!"
200 00013170 207265616479206F72- <1>
200 00013179 207772697465206572- <1>
200 00013182 726F7221 <1>
201 00013186 0D0A00 <1> db 0Dh, 0Ah, 0
202 <1>
203 <1> Msg_Dir_Not_Found:
204 00013189 4469726563746F7279- <1> db "Directory not found!"
204 00013192 206E6F7420666F756E- <1>
204 0001319B 6421 <1>
205 0001319D 0D0A00 <1> db 0Dh, 0Ah, 0
206 <1>
207 <1> Msg_File_Not_Found:
208 000131A0 46696C65206E6F7420- <1> db "File not found!"
208 000131A9 666F756E6421 <1>
209 000131AF 0D0A00 <1> db 0Dh, 0Ah, 0
210 <1>
211 <1> Msg_File_Directory_Not_Found:
212 000131B2 46696C65206F722064- <1> db "File or directory not found!"
212 000131BB 69726563746F727920- <1>
212 000131C4 6E6F7420666F756E64- <1>
212 000131CD 21 <1>
213 000131CE 0D0A00 <1> db 0Dh, 0Ah, 0
214 <1>
215 <1> Msg_LongName_Not_Found:
216 000131D1 4C6F6E67206E616D65- <1> db "Long name not found!"
216 000131DA 206E6F7420666F756E- <1>
216 000131E3 6421 <1>
217 000131E5 0D0A00 <1> db 0Dh, 0Ah, 0
218 <1>
219 <1> beep_Insufficient_Memory: ; 20/02/2017
220 000131E8 0D0A <1> db 0Dh, 0Ah
221 000131EA 07 <1> db 07h
222 <1> Msg_Insufficient_Memory:
223 000131EB 496E73756666696369- <1> db "Insufficient memory!"
223 000131F4 656E74206D656D6F72- <1>
223 000131FD 7921 <1>
224 000131FF 0D0A00 <1> db 0Dh, 0Ah, 0
225 <1>
226 <1> Msg_Error_Code:
227 00013202 436F6D6D616E642066- <1> db 'Command failed! Error code : '
227 0001320B 61696C656421204572- <1>
227 00013214 726F7220636F646520- <1>
227 0001321D 3A20 <1>
228 0001321F 303068 <1> error_code_hex: db '00h'
229 00013222 0D0A00 <1> db 0Dh, 0Ah, 0
230 <1>
231 <1> ; 19/12/2025
232 <1> ;align 2
233 <1>
234 <1> ; 10/02/2016
235 <1> ; DIR.ASM - 09/10/2011
236 <1>
237 00013225 3C4449523E20202020- <1> Type_Dir: db '<DIR>' ; 10 bytes
237 0001322E 20 <1>
238 <1>
239 <1> File_Name:
240 0001322F 20<rep Ch> <1> times 12 db 20h
241 0001323B 20 <1> db 20h
242 <1> Dir_Or_FileSize:
243 0001323C 20<rep Ah> <1> times 10 db 20h
244 00013246 20 <1> db 20h
245 <1> File_Attribute:
246 00013247 20202020 <1> dd 20202020h
247 0001324B 20 <1> db 20h
248 <1> File_Day:
249 0001324C 3030 <1> db '0','0'
250 0001324E 2F <1> db '/'
251 <1> File_Month:
252 0001324F 3030 <1> db '0','0'
253 00013251 2F <1> db '/'
254 <1> File_Year:
255 00013252 30303030 <1> db '0','0','0','0'
256 00013256 20 <1> db 20h
257 <1> File_Hour:
258 00013257 3030 <1> db '0','0'
259 00013259 3A <1> db ':'
260 <1> File_Minute:
261 0001325A 3030 <1> db '0','0'
262 0001325C 00 <1> db 0
263 <1>
264 <1> Decimal_File_Count_Header:
265 0001325D 0D0A <1> db 0Dh, 0Ah
266 <1> Decimal_File_Count:
267 0001325F 00<rep 6h> <1> times 6 db 0

```

```

268                                     <1>
269 00013265 2066696C6528732920- <1> str_files: db " file(s) & "
269 0001326E 2620 <1>
270                                     <1> Decimal_Dir_Count:
271 00013270 00<rep 6h> <1> times 6 db 0
272                                     <1> str_dirs:
273 00013276 206469726563746F72- <1> db " directory(s) "
273 0001327F 7928732920 <1>
274 00013284 0D0A00 <1> db 0Dh, 0Ah, 0
275                                     <1>
276 00013287 206279746528732920- <1> str_bytes: db " byte(s) in file(s)"
276 00013290 696E2066696C652873- <1>
276 00013299 29 <1>
277 0001329A 0D0A00 <1> db 0Dh, 0Ah, 0
278                                     <1>
279                                     <1> ; CMD_INTR.ASM - 09/11/2011
280                                     <1> ; 07/10/2010
281                                     <1> Msg_invalid_name_chars:
282 0001329D 496E76616C69642066- <1> db "Invalid file or directory name characters!"
282 000132A6 696C65206F72206469- <1>
282 000132AF 726563746F7279206E- <1>
282 000132B8 616D65206368617261- <1>
282 000132C1 637465727321 <1>
283 000132C7 0D0A00 <1> db 0Dh, 0Ah, 0
284                                     <1> ; 21/02/2016
285 000132CA 46696C65206F722064- <1> Msg_Name_Exists: db "File or directory name exists!"
285 000132D3 69726563746F727920- <1>
285 000132DC 6E616D652065786973- <1>
285 000132E5 747321 <1>
286 000132E8 0D0A00 <1> db 0Dh, 0Ah, 0
287                                     <1> Msg_DoYouWantMkdir:
288 000132EB 446F20796F75207761- <1> db "Do you want to make directory ", 0
288 000132F4 6E7420746F206D616B- <1>
288 000132FD 65206469726563746F- <1>
288 00013306 72792000 <1>
289 0001330A 2028592F4E29203F20- <1> Msg_YesNo: db " (Y/N) ? ", 0
289 00013313 00 <1>
290 00013314 000D0A00 <1> Y_N_nextline: db 0, 0Dh, 0Ah, 0
291 00013318 4F4B2E0D0A00 <1> Msg_OK: db "OK.", 0Dh, 0Ah, 0
292                                     <1>
293                                     <1> ; 27/02/2016
294                                     <1> Msg_DoYouWantRmdir:
295 0001331E 446F20796F75207761- <1> db "Do you want to delete directory ", 0
295 00013327 6E7420746F2064656C- <1>
295 00013330 657465206469726563- <1>
295 00013339 746F72792000 <1>
296                                     <1> Msg_Dir_Not_Empty:
297 0001333F 4469726563746F7279- <1> db "Directory not empty!"
297 00013348 206E6F7420656D7074- <1>
297 00013351 7921 <1>
298 00013353 0D0A00 <1> db 0Dh, 0Ah, 0
299                                     <1>
300                                     <1> Msg_DoYouWantDelete:
301 00013356 446F20796F75207761- <1> db "Do you want to delete file ",0
301 0001335F 6E7420746F2064656C- <1>
301 00013368 6574652066696C6520- <1>
301 00013371 00 <1>
302                                     <1>
303 00013372 44656C657465642E2E- <1> Msg_Deleted: db "Deleted...", 0Dh, 0Ah, 0
303 0001337B 2E0D0A00 <1>
304                                     <1>
305                                     <1> Msg_Permission_Denied:
306 0001337F 07 <1> db 7
307 00013380 5065726D697373696F- <1> db "Permission denied!", 0Dh, 0Ah, 0
307 00013389 6E2064656E69656421- <1>
307 00013392 0D0A00 <1>
308                                     <1>
309                                     <1> ; 04/03/2016
310 00013395 4E657720 <1> Msg_New: db "New "
311 00013399 00 <1> db 0
312                                     <1> Str_Attributes:
313 0001339A 417474726962757465- <1> db "Attributes : "
313 000133A3 73203A20 <1>
314 000133A7 4E4F524D414C <1> Attr_Chars: db "NORMAL"
315 000133AD 00 <1> db 0
316                                     <1>
317                                     <1> ; 06/03/2016
318                                     <1> ; CMD_INTR.ASM - 16/11/2010
319                                     <1> Msg_DoYouWantRename:
320 000133AE 446F20796F75207761- <1> db "Do you want to rename ", 0
320 000133B7 6E7420746F2072656E- <1>
320 000133C0 616D652000 <1>
321 000133C5 66696C652000 <1> Rename_File: db "file ", 0
322 000133CB 6469726563746F7279- <1> Rename_Directory: db "directory ", 0
322 000133D4 2000 <1>
323 000133D6 00<rep Dh> <1> Rename_OldName: times 13 db 0
324 000133E3 20617320 <1> Msg_File_rename_as: db " as "
325 000133E7 00<rep Dh> <1> Rename_NewName: times 13 db 0
326                                     <1>
327                                     <1> ; 08/03/2016
328                                     <1> ; CMD_INTR.ASM - 01/08/2010 - 23/04/2011
329                                     <1> msg_not_same_drv:
330 000133F4 4E6F742073616D6520- <1> db "Not same drive!"
330 000133FD 647269766521 <1>
331 00013403 0D0A00 <1> db 0Dh, 0Ah, 0
332                                     <1>
333                                     <1> Msg_DoYouWantMoveFile:
334 00013406 446F20796F75207761- <1> db "Do you want to move file", 0
334 0001340F 6E7420746F206D6F76- <1>
334 00013418 652066696C6500 <1>
335                                     <1>
336                                     <1> msg_insufficient_disk_space:
337 0001341F 496E73756666696369- <1> db "Insufficient disk space!"
337 00013428 656E74206469736B20- <1>
337 00013431 737061636521 <1>
338 00013437 0D0A00 <1> db 0Dh, 0Ah, 0
339                                     <1>
340                                     <1> ; 01/08/2010
341                                     <1> msg_source_file:
342 0001343A 0D0A536F7572636520- <1> db 0Dh, 0Ah, "Source file name : "
342 00013443 66696C65206E616D65- <1>
342 0001344C 2020202020203A2020- <1>
342 00013455 20 <1>
343                                     <1> msg_source_file_drv:
344 00013456 203A00 <1> db ":", 0
345                                     <1> msg_destination_file:
346 00013459 0D0A44657374696E61- <1> db 0Dh, 0Ah, "Destination file name : "
346 00013462 74696F6E2066696C65- <1>
346 0001346B 206E616D65203A2020- <1>
346 00013474 20 <1>
347                                     <1> msg_destination_file_drv:
348 00013475 203A00 <1> db ":", 0
349                                     <1> msg_copy_nextline:
350 00013478 0D0A00 <1> db 0Dh, 0Ah, 0

```

```

351 <1>
352 <1> ; 15/03/2016
353 <1> ; CMD_INTR.ASM
354 <1>
355 <1> Msg_DoYouWantOverWriteFile:
356 0001347B 446F20796F75207761- <1> db "Do you want to overwrite file ",0
356 00013484 6E7420746F206F7665- <1>
356 0001348D 727772697465206669- <1>
356 00013496 6C652000 <1>
357 <1>
358 <1> Msg_DoYouWantCopyFile:
359 0001349A 446F20796F75207761- <1> db "Do you want to copy file",0
359 000134A3 6E7420746F20636F70- <1>
359 000134AC 792066696C6500 <1>
360 <1>
361 <1> Msg_read_file_error_before_EOF:
362 000134B3 46696C652072656164- <1> db "File reading error! (before EOF)"
362 000134BC 696E67206572726F72- <1>
362 000134C5 2120286265666F7265- <1>
362 000134CE 20454F4629 <1>
363 000134D3 0A0A00 <1> db 0Ah, 0Ah, 0
364 <1>
365 <1> ; 18/03/2016
366 <1> ; TRDOS 386 (v2.0) mainprog copy procedure
367 <1> msg_reading:
368 000134D6 52656164696E672E2E- <1> db "Reading... ", 0
368 000134DF 2E2000 <1>
369 <1> msg_writing:
370 000134E2 57726974696E672E2E- <1> db "writing... ", 0
370 000134EB 2E2000 <1>
371 <1> percentagestr:
372 000134EE 2020202500 <1> db " %", 0 ; " 0%" .. "100%"
373 <1> ; 11/04/2016
374 <1> Msg_No_Set_Space:
375 000134F3 496E73756666696369- <1> db "Insufficient environment space!"
375 000134FC 656E7420656E766972- <1>
375 00013505 6F6E6D656E74207370- <1>
375 0001350E 61636521 <1>
376 00013512 0D0A00 <1> db 0Dh, 0Ah, 0
377 <1> ; 18/04/2016
378 <1> isc_msg:
379 00013515 0D0A <1> db 0Dh, 0Ah
380 00013517 494E56414C49442053- <1> db "INVALID SYSTEM CALL", 0
380 00013520 595354454D2043414C- <1>
380 00013529 4C00 <1>
381 <1> usi_msg:
382 0001352B 0D0A <1> db 0Dh, 0Ah
383 0001352D 554E444546494E4544- <1> db "UNDEFINED SOFTWARE INTERRUPT", 0
383 00013536 20534F465457415245- <1>
383 0001353F 20494E544552525550- <1>
383 00013548 5400 <1>
384 <1> ifc_msg:
385 0001354A 0D0A <1> db 0Dh, 0Ah
386 0001354C 494E56414C49442046- <1> db "INVALID FUNCTION CALL"
386 00013555 554E4354494F4E2043- <1>
386 0001355E 414C4C <1>
387 <1> inv_msg_for_trdos_v2:
388 00013561 20 <1> db 20h
389 00013562 666F72205452444F53- <1> db "for TRDOS v2!"
389 0001356B 20763221 <1>
390 0001356F 07 <1> db 07h
391 00013570 0D0A <1> db 0Dh, 0Ah
392 00013572 0D0A <1> db 0Dh, 0Ah
393 00013574 494E5420 <1> db "INT "
394 00013578 303068 <1> int_num_str: db "00h"
395 0001357B 0D0A <1> db 0Dh, 0Ah
396 0001357D 454158203A20 <1> db "EAX : "
397 00013583 303030303030303068- <1> eax_str: db "00000000h", 0Dh, 0Ah
397 0001358C 0D0A <1>
398 0001358E 454950203A20 <1> db "EIP : "
399 00013594 303030303030303068- <1> eip_str: db "00000000h", 0Dh, 0Ah, 0
399 0001359D 0D0A00 <1>
400 <1>
401 <1> ; 17/04/2021
402 <1> ; ('KDEV' device parameters are disabled as temporary)
403 <1>
404 <1> ; 07/10/2016
405 <1> ; Device names & parameters (for kernel devices)
406 <1>
407 <1> align 2
408 <1> ;KDEV_NAME:
409 <1> ; db 'TTY',0,0,0,0,0 ; 1
410 <1> ; db 'MEM',0,0,0,0,0 ; 2
411 <1> ; db 'FD0',0,0,0,0,0 ; 3
412 <1> ; db 'FD1',0,0,0,0,0 ; 4
413 <1> ; db 'HD0',0,0,0,0,0 ; 5
414 <1> ; db 'HD1',0,0,0,0,0 ; 6
415 <1> ; db 'HD2',0,0,0,0,0 ; 7
416 <1> ; db 'HD3',0,0,0,0,0 ; 8
417 <1> ; db 'LPT',0,0,0,0,0 ; 9
418 <1> ; db 'TTY0',0,0,0,0,0 ; 10
419 <1> ; db 'TTY1',0,0,0,0,0 ; 11
420 <1> ; db 'TTY2',0,0,0,0,0 ; 12
421 <1> ; db 'TTY3',0,0,0,0,0 ; 13
422 <1> ; db 'TTY4',0,0,0,0,0 ; 14
423 <1> ; db 'TTY5',0,0,0,0,0 ; 15
424 <1> ; db 'TTY6',0,0,0,0,0 ; 16
425 <1> ; db 'TTY7',0,0,0,0,0 ; 17
426 <1> ; db 'TTY8',0,0,0,0,0 ; 18
427 <1> ; db 'TTY9',0,0,0,0,0 ; 19
428 <1> ; db 'COM1',0,0,0,0,0 ; 18
429 <1> ; db 'COM2',0,0,0,0,0 ; 19
430 <1> ; db 'CONSOLE',0 ; 1
431 <1> ; db 'PRINTER',0 ; 9
432 <1> ; db 'CDROM' ; 20
433 <1> ; db 'CDROM0' ; 20
434 <1> ; db 'CDROM1' ; 21
435 <1> ; db 'DVD' ; 22
436 <1> ; db 'DVD0' ; 22
437 <1> ; db 'DVD1' ; 23
438 <1> ; db 'USB' ; 24
439 <1> ; db 'USB0' ; 24
440 <1> ; db 'USB1' ; 25
441 <1> ; db 'USB2' ; 26
442 <1> ; db 'USB3' ; 27
443 <1> ; db 'KEYBOARD' ; 1
444 <1> ; db 'MOUSE' ; 28
445 <1> ; db 'SOUND' ; 29
446 <1> ; db 'VGA',0,0,0,0,0 ; 30
447 <1> ; db 'CGA',0,0,0,0,0 ; 31
448 <1> ; db 'AUDIO',0,0,0,0,0 ; 29
449 <1> ; db 'VIDEO',0,0,0,0,0 ; 32
450 <1> ; db 'MUSIC',0,0,0,0,0 ; 33
451 <1> ; db 'ETHERNET' ; 34

```

```

452 <1> ; ;db 'SD0',0,0,0,0,0 ; 35
453 <1> ; ;db 'SD1',0,0,0,0,0 ; 36
454 <1> ; ;db 'SD2',0,0,0,0,0 ; 37
455 <1> ; ;db 'SD3',0,0,0,0,0 ; 38
456 <1> ; ;db 'SATA0' ; 35
457 <1> ; ;db 'SATA1' ; 36
458 <1> ; ;db 'SATA2' ; 37
459 <1> ; ;db 'SATA3' ; 38
460 <1> ; ;db 'PATA0',0,0,0 ; 5
461 <1> ; ;db 'PATA1',0,0,0 ; 6
462 <1> ; ;db 'PATA2',0,0,0 ; 7
463 <1> ; ;db 'PATA3',0,0,0 ; 8
464 <1> ; ;db 'WIRELESS' ; 39
465 <1> ; ;db 'HDMI',0,0,0,0 ; 40
466 <1> ; ;db 'NULL',0,0,0,0 ; 0
467 <1> ;
468 <1> ;NumOfKernelDevNames equ ($-KDEV_NAME) / 8 ; 20 (07/10/2016)
469 <1> ;
470 <1> ;KDEV_NUMBER:
471 <1> ; db 1,2,3,4,5,6,7,8,9
472 <1> ; db 10,11,12,13,14,15,16,17,18,19
473 <1> ; db 18,19,0
474 <1> ;
475 <1> ;NumOfKernelDevices equ $ - KDEV_NUMBER
476 <1> ;
477 <1> ;KDEV_OADDR:
478 <1> ; dd otty ;tty ; 1
479 <1> ; dd sret ;mem ; 2
480 <1> ; dd sret ;fd0 ; 3
481 <1> ; dd sret ;fd1 ; 4
482 <1> ; dd sret ;hd0 ; 5
483 <1> ; dd sret ;hd1 ; 6
484 <1> ; dd sret ;hd2 ; 7
485 <1> ; dd sret ;hd3 ; 8
486 <1> ; dd sret ;lpt ; 9
487 <1> ; dd ocvt ;tty0 ; 10
488 <1> ; dd ocvt ;tty1 ; 11
489 <1> ; dd ocvt ;tty2 ; 12
490 <1> ; dd ocvt ;tty3 ; 13
491 <1> ; dd ocvt ;tty4 ; 14
492 <1> ; dd ocvt ;tty5 ; 15
493 <1> ; dd ocvt ;tty6 ; 16
494 <1> ; dd ocvt ;tty7 ; 17
495 <1> ; dd ocvt ;tty8 ; 18
496 <1> ; dd ocvt ;tty9 ; 19
497 <1> ; ;dd ocvt ;com1 ; 18
498 <1> ; ;dd ocvt ;com2 ; 19
499 <1> ; dd sret ;null ; 20
500 <1> ;KDEV_CADDR:
501 <1> ; dd ctty ;tty ; 1
502 <1> ; dd cret ;mem ; 2
503 <1> ; dd cret ;fd0 ; 3
504 <1> ; dd cret ;fd1 ; 4
505 <1> ; dd cret ;hd0 ; 5
506 <1> ; dd cret ;hd1 ; 6
507 <1> ; dd cret ;hd2 ; 7
508 <1> ; dd cret ;hd3 ; 8
509 <1> ; dd cret ;lpt ; 9
510 <1> ; dd ocvt ;tty0 ; 10
511 <1> ; dd ccvt ;tty1 ; 11
512 <1> ; dd ccvt ;tty2 ; 12
513 <1> ; dd ccvt ;tty3 ; 13
514 <1> ; dd ccvt ;tty4 ; 14
515 <1> ; dd ccvt ;tty5 ; 15
516 <1> ; dd ccvt ;tty6 ; 16
517 <1> ; dd ccvt ;tty7 ; 17
518 <1> ; dd ccvt ;tty8 ; 18
519 <1> ; dd ccvt ;tty9 ; 19
520 <1> ; ;dd ccvt ;com1 ; 18
521 <1> ; ;dd ccvt ;com2 ; 19
522 <1> ; dd cret ;null ; 20
523 <1> ;
524 <1> ;KDEV_RADDR:
525 <1> ; dd rtty ;tty ; 1
526 <1> ; dd rmem ;mem ; 2
527 <1> ; dd rfd ;fd0 ; 3
528 <1> ; dd rfd ;fd1 ; 4
529 <1> ; dd rhd ;hd0 ; 5
530 <1> ; dd rhd ;hd1 ; 6
531 <1> ; dd rhd ;hd2 ; 7
532 <1> ; dd rhd ;hd3 ; 8
533 <1> ; dd rlppt ;lpt ; 9
534 <1> ; dd rcvt ;tty0 ; 10
535 <1> ; dd rcvt ;tty1 ; 11
536 <1> ; dd rcvt ;tty2 ; 12
537 <1> ; dd rcvt ;tty3 ; 13
538 <1> ; dd rcvt ;tty4 ; 14
539 <1> ; dd rcvt ;tty5 ; 15
540 <1> ; dd rcvt ;tty6 ; 16
541 <1> ; dd rcvt ;tty7 ; 17
542 <1> ; dd rcvt ;tty8 ; 18
543 <1> ; dd rcvt ;tty9 ; 19
544 <1> ; ;dd rcvt ;com1 ; 18
545 <1> ; ;dd rcvt ;com2 ; 19
546 <1> ; dd rnull ;null ; 20
547 <1> ;KDEV_WADDR:
548 <1> ; dd wtty ;tty ; 1
549 <1> ; dd wmem ;mem ; 2
550 <1> ; dd wfd ;fd0 ; 3
551 <1> ; dd wfd ;fd1 ; 4
552 <1> ; dd whd ;hd0 ; 5
553 <1> ; dd whd ;hd1 ; 6
554 <1> ; dd whd ;hd2 ; 7
555 <1> ; dd whd ;hd3 ; 8
556 <1> ; dd wlppt ;lpt ; 9
557 <1> ; dd xmtt ;tty0 ; 10
558 <1> ; dd xmtt ;tty1 ; 11
559 <1> ; dd xmtt ;tty2 ; 12
560 <1> ; dd xmtt ;tty3 ; 13
561 <1> ; dd xmtt ;tty4 ; 14
562 <1> ; dd xmtt ;tty5 ; 15
563 <1> ; dd xmtt ;tty6 ; 16
564 <1> ; dd xmtt ;tty7 ; 17
565 <1> ; dd xmtt ;tty8 ; 18
566 <1> ; dd xmtt ;tty9 ; 19
567 <1> ; ;dd xmtt ;com1 ; 18
568 <1> ; ;dd xmtt ;com2 ; 19
569 <1> ; dd wnull ;null ; 20
570 <1> ;
571 <1> ; DEV_ACCESS bits:
572 <1> ; ; bit 0 = accessable by normal users
573 <1> ; ; bit 1 = read access permission
574 <1> ; ; bit 2 = write access permission
575 <1> ; ; bit 3 = IOCTL permission to users

```



```

576      <1>      ; bit 4 = block device if it is set
577      <1>      ; bit 5 = 16 bit or 1024 byte data
578      <1>      ; bit 6 = 32 bit or 2048 byte data
579      <1>      ; bit 7 = installable device driver
580      <1>
581      <1> ;KDEV_ACCESS: ; 08/10/2016
582      <1> ;      db 00000111b ; tty, 1
583      <1> ;      db 00000111b ; mem, 2
584      <1> ;      db 10001111b ; fd0, 3
585      <1> ;      db 10001111b ; fd1, 4
586      <1> ;      db 10001111b ; hd0, 5
587      <1> ;      db 10001111b ; hd1, 6
588      <1> ;      db 10001111b ; hd2, 7
589      <1> ;      db 10001111b ; hd3, 8
590      <1> ;      db 00000111b ; lpt, 9
591      <1> ;      db 00000111b ; tty0, 10
592      <1> ;      db 00000111b ; tty1, 11
593      <1> ;      db 00000111b ; tty2, 12
594      <1> ;      db 00000111b ; tty3, 13
595      <1> ;      db 00000111b ; tty4, 14
596      <1> ;      db 00000111b ; tty5, 15
597      <1> ;      db 00000111b ; tty6, 16
598      <1> ;      db 00000111b ; tty7, 17
599      <1> ;      db 00000111b ; tty8, 18
600      <1> ;      db 00000111b ; tty9, 19
601      <1> ;      ;db 00000111b ; com1, 18
602      <1> ;      ;db 00000111b ; com2, 19
603      <1> ;      db 00000000b ; null, 0
604      <1>
605      <1> ; 07/10/2016
606      <1> ;NumOfInstallableDevices equ 8
607      <1> ;NUMIDEV equ NumOfInstallableDevices ; 8
608      <1> ;NUMOFDEVICES equ NumOfKernelDevices + NumOfInstallableDevices
609      <1>
610      <1> ; 26/02/2017
611      <1> ; IRQ callback (& signal Response Byte) service availability
612      <1> ; 'syscalbac'
613      <1> ; *****
614      <1> ; IRQ 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
615      <1> ; ---
616      <1> ; --- 00 00 00 01 02 03 00 04 00 05 06 07 08 09 00 00
617      <1> ; *****
618      <1> ; IRQenum:
619      <1> ;      db 0,0,0,1,2,3,0,4,0,5,6,7,8,9,0,0
620      <1>
621      <1> ; 28/08/2017
622      <1> ; 20/08/2017
623      <1> ; DMA Registers (for 'sysdma')
624      <1> ; 02/07/2017 (sb16mod.s)
625      <1> dma_adr: db 0,2,4,6,0C0h,0C4h,0C8h,0CCh
626      <1> dma_cnt: db 1,3,5,7,0C2h,0C6h,0CAh,0CEh
627      <1> dma_page: db 87h,83h,81h,82h,8Fh,8Bh,89h,8Ah ; 03/08/2017
628      <1> dma_mask: db 0Ah,0Ah,0Ah,0Ah,0D4h,0D4h,0D4h,0D4h
629      <1> dma_mod: db 0Bh,0Bh,0Bh,0Bh,0D6h,0D6h,0D6h,0D6h
630      <1> dma_flip: db 0Ch,0Ch,0Ch,0Ch,0D8h,0D8h,0D8h,0D8h
3441
3442
3443
3444      000135E0 E0810B00
3445
3446      000135E4 00000000
3447
3448      Align 4
3449      ; 15/04/2016
3450      ; TRDOS 386 (TRDOS v2.0)
3451
3452      ; 21/08/2014
3453      ilist:
3454      ;times 32 dd cpu_except ; INT 0 to INT 1Fh
3455      ;
3456      ; Exception list
3457      ; 25/08/2014
3458      000135E8 [1E0C0000] dd exc0 ; 0h, Divide-by-zero Error
3459      000135EC [250C0000] dd exc1
3460      000135F0 [2C0C0000] dd exc2
3461      000135F4 [330C0000] dd exc3
3462      000135F8 [370C0000] dd exc4
3463      000135FC [3B0C0000] dd exc5
3464      00013600 [3F0C0000] dd exc6 ; 06h, Invalid Opcode
3465      00013604 [430C0000] dd exc7
3466      00013608 [470C0000] dd exc8
3467      0001360C [4B0C0000] dd exc9
3468      00013610 [4F0C0000] dd exc10
3469      00013614 [530C0000] dd exc11
3470      00013618 [570C0000] dd exc12
3471      0001361C [5B0C0000] dd exc13 ; 0Dh, General Protection Fault
3472      00013620 [5F0C0000] dd exc14 ; 0Eh, Page Fault
3473      00013624 [630C0000] dd exc15
3474      00013628 [670C0000] dd exc16
3475      0001362C [6B0C0000] dd exc17
3476      00013630 [6F0C0000] dd exc18
3477      00013634 [730C0000] dd exc19
3478      00013638 [770C0000] dd exc20
3479      0001363C [7B0C0000] dd exc21
3480      00013640 [7F0C0000] dd exc22
3481      00013644 [830C0000] dd exc23
3482      00013648 [870C0000] dd exc24
3483      0001364C [8B0C0000] dd exc25
3484      00013650 [8F0C0000] dd exc26
3485      00013654 [930C0000] dd exc27
3486      00013658 [970C0000] dd exc28
3487      0001365C [9B0C0000] dd exc29
3488      00013660 [9F0C0000] dd exc30
3489      00013664 [A30C0000] dd exc31
3490      IRQ_list: ; 28/02/2017 ('syscalbac')
3491      ; Interrupt list
3492      00013668 [88090000] dd timer_int ; INT 20h
3493      ;dd irq0
3494      0001366C [ED100000] dd kb_int ; 24/01/2016
3495      ;dd irq1
3496      00013670 [740B0000] dd irq2
3497      ; COM2 int
3498      00013674 [780B0000] dd irq3
3499      ; COM1 int
3500      00013678 [830B0000] dd irq4
3501      0001367C [8E0B0000] dd irq5
3502      ;DISKETTE_INT: ;06/02/2015
3503      00013680 [A04E0000] dd fdc_int ; 16/02/2015, IRQ 6 handler
3504      ;dd irq6
3505      ; Default IRQ 7 handler against spurious IRQs (from master PIC)
3506      ; 25/02/2015 (source: http://wiki.osdev.org/8259_PIC)
3507      00013684 [FA0E0000] dd default_irq7 ; 25/02/2015
3508      ;dd irq7

```

```

3509 ; Real Time Clock Interrupt
3510 00013688 [FC0A0000] dd rtc_int ; 23/02/2015, IRQ 8 handler
3511 ;dd irq8 ; INT 28h
3512 0001368C [9E0B0000] dd irq9
3513 00013690 [A20B0000] dd irq10
3514 00013694 [A60B0000] dd irq11
3515 00013698 [AA0B0000] dd irq12
3516 0001369C [AE0B0000] dd irq13
3517 ;HDISK_INT1: ;06/02/2015
3518 000136A0 [DF570000] dd hdc1_int ; 21/02/2015, IRQ 14 handler
3519 ;dd irq14
3520 ;HDISK_INT2: ;06/02/2015
3521 000136A4 [02580000] dd hdc2_int ; 21/02/2015, IRQ 15 handler
3522 ;dd irq15 ; INT 2Fh
3523 ; 14/08/2015
3524 ;dd sysent ; INT 30h (system calls)
3525 ;
3526 ; 15/04/2016
3527 ; TRDOS 386(TRDOS v2.0) Software Interrupts
3528
3529 000136A8 [05370100] dd int30h ; Reserved for
3530 ; !!! Retro UNIX (RUNIX) !!!
3531 ; !!! SINGLIX !!! System Calls
3532 000136AC [7C170000] dd int31h ; Video BIOS (IBM PC/AT, Int 10h)
3533 000136B0 [490F0000] dd int32h ; Keyboard Functions (IBM PC/AT, Int 16h)
3534 000136B4 [424F0000] dd int33h ; DISK I/O (IBM PC/AT, Int 13h)
3535 000136B8 [8C1B0100] dd int34h ; #IOCTL# (I/O port access support for ring 3)
3536 000136BC [E5620000] dd int35h ; Time/Date Functions (IBM PC/AT, Int 1Ah)
3537 000136C0 [AE0D0000] dd ignore_int ; INT 36h : Timer Functions
3538 000136C4 [AE0D0000] dd ignore_int ; INT 37h
3539 000136C8 [AE0D0000] dd ignore_int ; INT 38h
3540 000136CC [AE0D0000] dd ignore_int ; INT 39h
3541 000136D0 [AE0D0000] dd ignore_int ; INT 3Ah
3542 000136D4 [AE0D0000] dd ignore_int ; INT 3Bh
3543 000136D8 [AE0D0000] dd ignore_int ; INT 3Ch
3544 000136DC [AE0D0000] dd ignore_int ; INT 3Dh
3545 000136E0 [AE0D0000] dd ignore_int ; INT 3Eh
3546 000136E4 [AE0D0000] dd ignore_int ; INT 3Fh
3547 000136E8 [C6C80000] dd sysent ; INT 40h : !!! TRDOS 386 System Calls !!!
3548 ;dd ignore_int
3549 000136EC 00000000 dd 0
3550
3551 ; 20/08/2014
3552 ; /* This is the default interrupt "handler" :-) */
3553 ; Linux v0.12 (head.s)
3554 int_msg:
3555 000136F0 556E6B6E6F776E2069- db "Unknown interrupt ! ", 0
3556 000136F9 6E7465727275707420-
3557 00013702 212000
3558
3559 ; 15/04/2016
3560 ; TRDOS 386 (TRDOS v2.0)
3561
3562 ; 29/04/2016
3563 int30h:
3564 trdos_isc_routine:
3565 ; 02/05/2016
3566 ; 01/05/2016
3567 ; 29/04/2016
3568 ; 18/04/2016
3569 ; 15/04/2016 (TRDOS 386 = TRDOS v2.0)
3570 ; 17/04/2011 (TRDOS v1.0, 'IFC.ASM')
3571 ; 03/02/2011 ('trdos_ifc_routine')
3572 ;
3573 mov ebx, [esp] ; EIP (next)
3574 sub ebx, 2 ; EIP (CD ##h)
3575
3576 mov ecx, eax
3577 mov al, [ebx+1] ; CDh ##h
3578
3579 mov dx, KDATA
3580 mov ds, dx
3581 mov es, dx
3582
3583 cld
3584 mov edx, [k_page_dir]
3585 mov cr3, edx
3586
3587 call byteto hex
3588 mov [int_num_str], ax
3589
3590 mov eax, ebx ; EIP
3591 call dwordto hex
3592 mov [eip_str], edx
3593 mov [eip_str+4], eax
3594
3595 mov eax, ecx
3596 call dwordto hex
3597 mov [eax_str], edx
3598 mov [eax_str+4], eax
3599
3600 inc ebx
3601 mov al, [ebx] ; Interrupt number
3602
3603 trdos_isc_handler:
3604 cmp dh, 30h ; Retro UNIX, SINGLIX System calls
3605 jne short trdos_usi_handler
3606 mov esi, isc_msg
3607 jmp short loc_write_inv_system_call_msg
3608
3609 trdos_usi_handler:
3610 mov esi, usi_msg
3611
3612 loc_write_inv_system_call_msg:
3613 call print_msg
3614 ; 29/04/2016
3615 mov esi, inv_msg_for_trdos_v2
3616 call print_msg
3617
3618 loc_ifc_terminate_process:
3619 ; u.uno = process number
3620 ; 29/04/2016
3621 ; 02/05/2016
3622 inc byte [sysflg] ; 0FFh -> 0
3623
3624 mov eax, 1
3625 jmp sysexit
3626
3627 ; 07/03/2015
3628 ; Temporary Code
3629 display_disks:
3630 cmp byte [fd0_type], 0
3631 jna short ddsks1

```

```

3631 0001378D E87D000000      call    pdskm
3632
3633 00013792 803D[7D660000]00    ddsks1:
3634 00013799 760C                cmp     byte [fd1_type], 0
3635 0001379B C605[DF380100]31          jna     short ddsks2
3636 000137A2 E868000000          mov     byte [dskx], '1'
3637                                call    pdskm
3638 000137A7 803D[7E660000]00    ddsks2:
3639 000137AE 7654                cmp     byte [hd0_type], 0
3640 000137B0 66C705[DD380100]68-   jna     short ddsks6
3640 000137B8 64                mov     word [dsktype], 'hd'
3641 000137B9 C605[DF380100]30          mov     byte [dskx], '0'
3642 000137C0 E84A000000          call    pdskm
3643
3644 000137C5 803D[7F660000]00    ddsks3:
3645 000137CC 7636                cmp     byte [hd1_type], 0
3646 000137CE C605[DF380100]31          jna     short ddsks6
3647 000137D5 E835000000          mov     byte [dskx], '1'
3648                                call    pdskm
3649 000137DA 803D[80660000]00    ddsks4:
3650 000137E1 7621                cmp     byte [hd2_type], 0
3651 000137E3 C605[DF380100]32          jna     short ddsks6
3652 000137EA E820000000          mov     byte [dskx], '2'
3653                                call    pdskm
3654 000137EF 803D[81660000]00    ddsks5:
3655 000137F6 760C                cmp     byte [hd3_type], 0
3656 000137F8 C605[DF380100]33          jna     short ddsks6
3657 000137FF E80B000000          mov     byte [dskx], '3'
3658                                call    pdskm
3659 00013804 BE[07390100]          ddsks6:
3660 00013809 E806000000          mov     esi, nextline
3661                                call    pdskm1
3662 0001380E C3                pdskm_ok:
3663                                retn
3664 0001380F BE[DB380100]          pdskm:
3665                                mov     esi, dsk_ready_msg
3666 00013814 AC                pdskm1:
3667 00013815 08C0            lodsb
3668 00013817 74F5            or     al, al
3669 00013819 56                jz     short pdskm_ok
3670                                push    esi
3671 0001381A B807000000          ; 13/05/2016
3672                                mov     ebx, 7 ; Black background,
3673                                ; light gray forecolor
3674 0001381F E8B7EAFEFF          ; video page 0 (bh=0)
3675 00013824 5E                call    _write_tty
3676 00013825 EBED            pop     esi
3677                                jmp     short pdskm1
3678 00013827 90                Align 2
3679                                ; 21/08/2014
3680                                exc_msg:
3681 00013828 435055206578636570-   db "CPU exception ! "
3682 00013831 74696F6E202120
3683 00013838 3F3F68202045495020-   excnstr:
3684 00013841 3A20                db "??h", " EIP : "
3685 00013843 00<rep Ch>          EIPstr: ; 29/08/2014
3686                                times 12 db 0
3687                                ; 23/02/2015
3688                                ; 25/08/2014
3689                                ;scounter:
3690                                ; db 5
3691                                ; db 19
3692
3693                                ; 06/11/2014
3694                                ; Memory Information message
3695                                ; 14/08/2015
3696                                msg_memory_info:
3697 0001384F 07                db 07h
3698 00013850 0D0A            db 0Dh, 0Ah
3699                                ;db
3700 00013852 546F74616C206D656D-   db "MEMORY ALLOCATION INFO", 0Dh, 0Ah, 0Dh, 0Ah
3701 0001385B 6F7279203A20      db "Total memory : "
3702 00013861 303030303030303030-   mem_total_b_str: ; 10 digits
3702 0001386A 302062797465730D0A   db "0000000000 bytes", 0Dh, 0Ah
3703 00013873 202020202020202020-   db " ", 20h, 20h, 20h
3703 0001387C 202020202020202020
3704
3705 00013885 303030303030302070-   mem_total_p_str: ; 7 digits
3705 0001388E 616765730D0A         db "0000000 pages", 0Dh, 0Ah
3706 00013894 0D0A            db 0Dh, 0Ah
3707 00013896 46726565206D656D6F-   db "Free memory : "
3707 0001389F 727920203A20
3708
3709 000138A5 3F3F3F3F3F3F3F3F3F-   free_mem_b_str: ; 10 digits
3709 000138AE 3F2062797465730D0A   db "????????? bytes", 0Dh, 0Ah
3710 000138B7 202020202020202020-   db " ", 20h, 20h, 20h
3710 000138C0 202020202020202020
3711
3712 000138C9 3F3F3F3F3F3F3F2070-   free_mem_p_str: ; 7 digits
3712 000138D2 616765730D0A         db "??????? pages", 0Dh, 0Ah
3713 000138D8 0D0A00          db 0Dh, 0Ah, 0
3714
3715                                dsk_ready_msg:
3716 000138DB 0D0A            db 0Dh, 0Ah
3717                                dsktype:
3718 000138DD 6664            db 'fd'
3719                                dskx:
3720 000138DF 30                db '0'
3721 000138E0 20                db 20h
3722 000138E1 697320524541445920-   db 'is READY ...'
3722 000138EA 2E2E2E          db 0
3723 000138ED 00
3724
3725                                setup_error_msg:
3726 000138EE 0D0A            db 0Dh, 0Ah
3727 000138F0 4469736B2053657475-   db 'Disk Setup Error !'
3727 000138F9 70204572726F722021
3728 00013902 0D0A00          db 0Dh, 0Ah, 0
3729
3730                                next2line: ; 08/02/2016
3731 00013905 0D0A            db 0Dh, 0Ah
3732                                nextline:
3733 00013907 0D0A00          db 0Dh, 0Ah, 0
3734
3735                                ; temporary
3736                                ; 19/12/2020
3737                                msg_lfb_addr:
3738                                ;db 0Dh, 0Ah
3739 0001390A 4C696E656172206672-   db "Linear frame buffer at "
3739 00013913 616D65206275666665-
3739 0001391C 7220617420

```

```

3740 lfb_addr_str: ; 8 (hex) digits
3741 00013921 303030303030303068- db "00000000h", 0Dh, 0Ah
3741 0001392A 0D0A
3742 0001392C 0D0A00 db 0Dh, 0Ah, 0
3743
3744 ; KERNEL - SYSINIT Messages
3745 ; 24/08/2015
3746 ; 13/04/2015 - (Retro UNIX 386 v1 Beginning)
3747 ; 14/07/2013
3748 ;kernel_init_err_msg:
3749 ; db 0Dh, 0Ah
3750 ; db 07h
3751 ; db 'Kernel initialization ERROR !'
3752 ; db 0Dh, 0Ah, 0
3753
3754 ;welcome_msg:
3755 ; db 0Dh, 0Ah
3756 ; db 07h
3757 ; db 'Welcome to TRDOS 386 Operating System !'
3758 ; db 0Dh, 0Ah
3759 ; db 'by Erdogan Tan - 31/12/2017 (v2.0.0)'
3760 ; db 0Dh, 0Ah, 0
3761
3762 panic_msg:
3763 0001392F 0D0A07 db 0Dh, 0Ah, 07h
3764 00013932 4552524F523A204B65- db 'ERROR: Kernel Panic !'
3764 0001393B 726E656C2050616E69-
3764 00013944 632021
3765 00013947 0D0A00 db 0Dh, 0Ah, 0
3766
3767 ;msg1_drv_not_ready:
3768 ; db 07h, 0Dh, 0Ah
3769 ; db 'Drive not ready or read error !'
3770 ; db 0Dh, 0Ah, 0
3771
3772 starting_msg:
3773 ;;;;db "Turkish Rational DOS v2.0 [18/04/2021] ...", 0
3774 ;;;;db "Turkish Rational DOS v2.0 [11/08/2022] ...", 0
3775 ;;;;db "Turkish Rational DOS v2.0 [30/08/2023] ...", 0
3776 ;;;;db "Turkish Rational DOS v2.0 [07/12/2023] ...", 0
3777 ;;;;db "Turkish Rational DOS v2.0 [23/06/2024] ...", 0
3778 ;;;;db "Turkish Rational DOS v2.0 [29/12/2024] ...", 0
3779 ;;;;db "Turkish Rational DOS v2.0 [28/01/2025] ...", 0
3780 ;;;;db "Turkish Rational DOS v2.0 [28/12/2025] ...", 0
3781 0001394A 5475726B6973682052- db "Turkish Rational DOS v2.0 [10/01/2026] ...", 0
3781 00013953 6174696F6E616C2044-
3781 0001395C 4F532076322E30205B-
3781 00013965 31302F30312F323032-
3781 0001396E 365D202E2E2E00
3782
3783 NextLine:
3784 00013975 0D0A00 db 0Dh, 0Ah, 0
3785
3786 %include 'audio.s' ; 03/04/2017
1 <1> ; *****
2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.10 - audio.s
3 <1> ;
4 <1> ; Last Update: 28/01/2025 (Previous: 06/06/2024 - Kernel v2.0.9)
5 <1> ;
6 <1> ; Beginning: 03/04/2017
7 <1> ;
8 <1> ; Assembler: NASM version 2.15 (trdos386.s)
9 <1> ; *****
10 <1>
11 <1> ; AUDIO CONTROLLER & CODEC DEFINITIONS & CODE FOR TRDOS 386
12 <1>
13 <1> ;=====
14 <1> ; EQUATES
15 <1> ;=====
16 <1>
17 <1> ; PCI EQUATES
18 <1>
19 <1> BIT0 EQU 1
20 <1> BIT1 EQU 2
21 <1> BIT2 EQU 4
22 <1> BIT3 EQU 8
23 <1> BIT4 EQU 10h
24 <1> BIT5 EQU 20h
25 <1> BIT6 EQU 40h
26 <1> BIT7 EQU 80h
27 <1> BIT8 EQU 100h
28 <1> BIT9 EQU 200h
29 <1> BIT10 EQU 400h
30 <1> BIT11 EQU 800h
31 <1> BIT12 EQU 1000h
32 <1> BIT13 EQU 2000h
33 <1> BIT14 EQU 4000h
34 <1> BIT15 EQU 8000h
35 <1> BIT16 EQU 10000h
36 <1> BIT17 EQU 20000h
37 <1> BIT18 EQU 40000h
38 <1> BIT19 EQU 80000h
39 <1> BIT20 EQU 100000h
40 <1> BIT21 EQU 200000h
41 <1> BIT22 EQU 400000h
42 <1> BIT23 EQU 800000h
43 <1> BIT24 EQU 1000000h
44 <1> BIT25 EQU 2000000h
45 <1> BIT26 EQU 4000000h
46 <1> BIT27 EQU 8000000h
47 <1> BIT28 EQU 10000000h
48 <1> BIT29 EQU 20000000h
49 <1> BIT30 EQU 40000000h
50 <1> BIT31 EQU 80000000h
51 <1> NOT_BIT31 EQU 7FFFFFFFh
52 <1>
53 <1> ; PCI equates
54 <1> ; PCI function address (PFA)
55 <1> ; bit 31 = 1
56 <1> ; bit 23:16 = bus number (0-255)
57 <1> ; bit 15:11 = device number (0-31)
58 <1> ; bit 10:8 = function number (0-7)
59 <1> ; bit 7:0 = register number (0-255)
60 <1>
61 <1> IO_ADDR_MASK EQU 0FFFEh ; mask off bit 0 for reading BARS
62 <1> PCI_INDEX_PORT EQU 0CF8h
63 <1> PCI_DATA_PORT EQU 0CFCh
64 <1> PCI32 EQU BIT31 ; bitflag to signal 32bit access
65 <1> PCI16 EQU BIT30 ; bitflag for 16bit access
66 <1> NOT_PCI32_PCI16 EQU 03FFFFFFh ; NOT BIT31+BIT30 ; 19/03/2017
67 <1>
68 <1> PCI_FN0 EQU 0 << 8
69 <1> PCI_FN1 EQU 1 << 8
70 <1> PCI_FN2 EQU 2 << 8

```

```

71 <1> PCI_FN3 EQU 3 << 8
72 <1> PCI_FN4 EQU 4 << 8
73 <1> PCI_FN5 EQU 5 << 8
74 <1> PCI_FN6 EQU 6 << 8
75 <1> PCI_FN7 EQU 7 << 8
76 <1>
77 <1> PCI_CMD_REG EQU 04h ; reg 04, command reg
78 <1> IO_ENA EQU BIT0 ; i/o decode enable
79 <1> MEM_ENA EQU BIT1 ; memory decode enable
80 <1> BM_ENA EQU BIT2 ; bus master enable
81 <1>
82 <1> ; VIA VT8233 EQUATES
83 <1>
84 <1> VIA_VID equ 1106h ; VIA's PCI vendor ID
85 <1> VT8233_DID equ 3059h ; VT8233 (VT8235) device ID
86 <1>
87 <1> PCI_IO_BASE equ 10h
88 <1> AC97_INT_LINE equ 3ch
89 <1> VIA_ACLINK_CTRL equ 41h
90 <1> VIA_ACLINK_STAT equ 40h
91 <1> VIA_ACLINK_C00_READY equ 01h ; primary codec ready
92 <1>
93 <1> VIA_REG_AC97 equ 80h ; dword
94 <1>
95 <1> VIA_ACLINK_CTRL_ENABLE equ 80h ; 0: disable, 1: enable
96 <1> VIA_ACLINK_CTRL_RESET equ 40h ; 0: assert, 1: de-assert
97 <1> VIA_ACLINK_CTRL_SYNC equ 20h ; 0: release SYNC, 1: force SYNC hi
98 <1> VIA_ACLINK_CTRL_VRA equ 08h ; 0: disable VRA, 1: enable VRA
99 <1> VIA_ACLINK_CTRL_PCM equ 04h ; 0: disable PCM, 1: enable PCM
100 <1> ; 3D Audio Channel slots 3/4
101 <1> VIA_ACLINK_CTRL_INIT equ (VIA_ACLINK_CTRL_ENABLE + VIA_ACLINK_CTRL_RESET +
VIA_ACLINK_CTRL_PCM + VIA_ACLINK_CTRL_VRA)
102 <1>
103 <1> CODEC_AUX_VOL equ 04h
104 <1> VIA_REG_AC97_BUSY equ 01000000h ; (1<<24)
105 <1> VIA_REG_AC97_CMD_SHIFT equ 10h ; 16
106 <1> VIA_REG_AC97_PRIMARY_VALID equ 02000000h ; (1<<25)
107 <1> VIA_REG_AC97_READ equ 00800000h ; (1<<23)
108 <1> VIA_REG_AC97_CODEC_ID_SHIFT equ 1Eh ; 30
109 <1> VIA_REG_AC97_CODEC_ID_PRIMARY equ 0
110 <1> VIA_REG_AC97_DATA_SHIFT equ 0
111 <1> VIADEV_PLAYBACK equ 0
112 <1> VIA_REG_OFFSET_STATUS equ 0 ; ; byte - channel status
113 <1> VIA_REG_OFFSET_CONTROL equ 01h ; ; byte - channel control
114 <1> VIA_REG_CTRL_START equ 80h ; ; WO
115 <1> VIA_REG_CTRL_TERMINATE equ 40h ; ; WO
116 <1> VIA_REG_CTRL_PAUSE equ 08h ; ; RW
117 <1> VIA_REG_CTRL_RESET equ 01h ; ; RW - probably reset? undocumented
118 <1> VIA_REG_OFFSET_STOP_IDX equ 08h ; ; dword - stop index, channel type, sample rate
119 <1> VIA8233_REG_TYPE_16BIT equ 200000h ; ; RW
120 <1> VIA8233_REG_TYPE_STEREO equ 100000h ; ; RW
121 <1> VIA_REG_OFFSET_CURR_INDEX equ 0Fh ; ; byte - channel current index (for via8233 only)
122 <1> VIA_REG_OFFSET_TABLE_PTR equ 04h ; ; dword - channel table pointer
123 <1> VIA_REG_OFFSET_CURR_PTR equ 04h ; ; dword - channel current pointer
124 <1> VIA_REG_OFS_PLAYBACK_VOLUME_L equ 02h ; ; byte
125 <1> VIA_REG_OFS_PLAYBACK_VOLUME_R equ 03h ; ; byte
126 <1> VIA_REG_CTRL_AUTOSTART equ 20h
127 <1> VIA_REG_CTRL_INT_EOL equ 02h
128 <1> VIA_REG_CTRL_INT_FLAG equ 01h
129 <1> VIA_REG_CTRL_INT equ (VIA_REG_CTRL_INT_FLAG + VIA_REG_CTRL_INT_EOL +
VIA_REG_CTRL_AUTOSTART)
130 <1>
131 <1>
132 <1> VIA_REG_STAT_STOP_IDX equ 10h ; ; RO ; 27/07/2020
133 <1> ; current index = stop index
134 <1> VIA_REG_STAT_STOPPED equ 04h ; ; RWC
135 <1> VIA_REG_STAT_EOL equ 02h ; ; RWC
136 <1> VIA_REG_STAT_FLAG equ 01h ; ; RWC
137 <1> VIA_REG_STAT_ACTIVE equ 80h ; ; RO
138 <1> ; 28/11/2016
139 <1> VIA_REG_STAT_LAST equ 40h ; ; RO
140 <1> VIA_REG_STAT_TRIGGER_QUEUED equ 08h ; ; RO
141 <1> VIA_REG_CTRL_INT_STOP equ 04h ; ; Interrupt on Current Index = Stop Index
142 <1> ; and End of Block
143 <1>
144 <1> VIA_REG_OFFSET_CURR_COUNT equ 0Ch ; ; dword - channel current count, index
145 <1>
146 <1> PORTB EQU 061h
147 <1> REFRESH_STATUS EQU 010h ; Refresh signal status
148 <1>
149 <1> ; AC97 Codec registers.
150 <1>
151 <1> ; 22/07/2020
152 <1> ; REALTEK ALC655 and ADI SOUNDMAX AD1980 CODEC MIXER REGISTERS
153 <1>
154 <1> ; each codec/mixer register is 16bits
155 <1>
156 <1> CODEC_RESET_REG equ 00h ; reset codec
157 <1> CODEC_MASTER_VOL_REG equ 02h ; master volume
158 <1> CODEC_HP_VOL_REG equ 04h ; headphone volume ; AD1980
159 <1> CODEC_MASTER_MONO_VOL_REG equ 06h ; master mono volume (mono-out)
160 <1> ; CODEC_MASTER_TONE_REG equ 08h ; master tone (R+L) ; (not used)
161 <1> CODEC_PCBEEP_VOL_REG equ 0Ah ; PC beep volume ; ALC655
162 <1> CODEC_PHONE_VOL_REG equ 0Ch ; phone volume
163 <1> CODEC_MIC_VOL_REG equ 0Eh ; mic volume
164 <1> CODEC_LINE_IN_VOL_REG equ 10h ; line in volume
165 <1> CODEC_CD_VOL_REG equ 12h ; CD volume
166 <1> ; CODEC_VID_VOL_REG equ 14h ; video volume ; (not used)
167 <1> CODEC_AUX_VOL_REG equ 16h ; aux volume
168 <1> CODEC_PCM_OUT_REG equ 18h ; PCM out volume
169 <1> CODEC_RECORD_SELECT_REG equ 1Ah ; record select
170 <1> CODEC_RECORD_VOL_REG equ 1Ch ; record volume (record gain)
171 <1> ; CODEC_RECORD_MIC_VOL_REG equ 1Eh ; record mic volume ; (not used)
172 <1> CODEC_GP_REG equ 20h ; general purpose
173 <1> ; CODEC_3D_CONTROL_REG equ 22h ; 3D control
174 <1> ; ; CODEC_AUDIO_INT_PAGING_REG equ 24h ; audio int & paging ; (not used)
175 <1> CODEC_POWER_CTRL_REG equ 26h ; power down control
176 <1> CODEC_EXT_AUDIO_REG equ 28h ; extended audio ID
177 <1> CODEC_EXT_AUDIO_CTRL_REG equ 2Ah ; extended audio status/control
178 <1> CODEC_PCM_FRONT_DACRATE_REG equ 2Ch ; PCM front sample rate
179 <1> CODEC_PCM_SURND_DACRATE_REG equ 2Eh ; PCM surround sample rate
180 <1> CODEC_PCM_LFE_DACRATE_REG equ 30h ; PCM Center/LFE sample rate
181 <1> CODEC_LR_ADCRATE_REG equ 32h ; PCM input sample rate
182 <1> CODEC_MIC_ADCRATE_REG equ 34h ; mic in sample rate ; AD1980
183 <1> CODEC_PCM_LFE_VOL_REG equ 36h ; PCM Center/LFE volume
184 <1> CODEC_PCM_SURND_VOL_REG equ 38h ; PCM surround volume
185 <1> ; CODEC_SPDIF_CTRL_REG equ 3Ah ; S/PDIF control
186 <1> ; 22/07/2020
187 <1> CODEC_MISC_CTRL_BITS_REG equ 76h ; misc control bits ; AD1980
188 <1> ;
189 <1> CODEC_VENDOR_ID1 equ 7Ch ; REALTEK: 414Ch, ADI: 4144h
190 <1> CODEC_VENDOR_ID2 equ 7Eh ; REALTEK: 4760h, ADI: 5370h
191 <1>
192 <1> ; VT8233 SGD bits (21/04/2017)
193 <1> FLAGEQU BIT30
194
195
196
197

```

```

198 <1> EOL EQU BIT31
199 <1>
200 <1> ; INTEL ICH EQUATES
201 <1> ; 28/05/2017
202 <1> INTEL_VID equ 8086h ; Intel's PCI vendor ID
203 <1> ; 20/11/2023
204 <1> ; (playwav2.com, ac97.inc, Erdogan Tan, 11/11/2023)
205 <1> ; 03/11/2023 - Erdogan Tan (Ref: MenuetOS AC97 WAV Player source code, 2004)
206 <1> SIS_VID equ 1039h
207 <1> NVIDIA_VID equ 10DEh ; Ref: MPXPLAY/SBEMU/KOLIBRIOS AC97 source c.
208 <1> AMD_VID equ 1022h
209 <1>
210 <1> ICH_DID equ 2415h ; ICH (82801AA) device ID
211 <1> ; 20/11/2023
212 <1> ; (playwav2.com, ac97.inc, Erdogan Tan, 11/11/2023)
213 <1> ; 17/02/2017 (Erdogan Tan, ref: ALSA Device IDs, ALSA project)
214 <1> ICH0_DID equ 2425h ; ICH0
215 <1> ICH2_DID equ 2445h ; ICH2
216 <1> ICH3_DID equ 2485h ; ICH3
217 <1> ICH4_DID equ 24C5h ; ICH4
218 <1> ICH5_DID equ 24D5h ; ICH5
219 <1> ICH6_DID equ 266Eh ; ICH6
220 <1> ESB6300_DID equ 25A6h ; 6300ESB
221 <1> ESB631X_DID equ 2698h ; 631XESB
222 <1> ICH7_DID equ 27DEh ; ICH7
223 <1> ; 03/11/2023 - Erdogan Tan (Ref: MenuetOS AC97 WAV Player source code, 2004)
224 <1> MX82440_DID equ 7195h
225 <1> SI7012_DID equ 7012h
226 <1> NFORCE_DID equ 01B1h
227 <1> NFORCE2_DID equ 006Ah
228 <1> AMD8111_DID equ 746Dh
229 <1> AMD768_DID equ 7445h
230 <1> ; 03/11/2023 - Erdogan Tan - Ref: MPXPLAY/SBEMU/KOLIBRIOS AC97 source code
231 <1> CK804_DID equ 0059h ; NFORCE4
232 <1> MCP04_DID equ 003Ah
233 <1> CK8_DID equ 008Ah
234 <1> NFORCE3_DID equ 00DAh
235 <1> CK8S_DID equ 00EAh
236 <1>
237 <1> NAMBAR_REG equ 10h ; native audio mixer Base Address Register
238 <1> NABMBAR_REG equ 14h ; native audio bus mastering Base Addr Reg
239 <1>
240 <1> PI_CR_REG equ 0Bh ; PCM in Control Register
241 <1> PO_CR_REG equ 1Bh ; PCM out Control Register
242 <1> MC_CR_REG equ 2Bh ; MIC in Control Register
243 <1>
244 <1> PI_SR_REG equ 6 ; PCM in Status register
245 <1> PO_SR_REG equ 16h ; PCM out Status register
246 <1> MC_SR_REG equ 26h ; MIC in Status register
247 <1>
248 <1> IOCE equ BIT4 ; interrupt on complete enable.
249 <1> FEIFE equ BIT3 ; set if you want an interrupt to fire
250 <1> LVBIIE equ BIT2 ; last valid buffer interrupt enable.
251 <1> RR equ BIT1 ; reset registers. Nukes all regs
252 <1> ; except bits 4:2 of this register.
253 <1> ; Only set this bit if BIT 0 is 0
254 <1> RPBm equ BIT0 ; Run/Pause
255 <1> ; set this bit to start the codec!
256 <1>
257 <1> PI_BDBAR_REG equ 0 ; PCM in buffer descriptor BAR
258 <1> PO_BDBAR_REG equ 10h ; PCM out buffer descriptor BAR
259 <1> MC_BDBAR_REG equ 20h ; MIC in buffer descriptor BAR
260 <1>
261 <1> PI_CIV_REG equ 4 ; PCM in current Index value (RO)
262 <1> PO_CIV_REG equ 14h ; PCM out current Index value (RO)
263 <1> MC_CIV_REG equ 24h ; MIC in current Index value (RO)
264 <1>
265 <1> PI_LVI_REG equ 5 ; PCM in Last Valid Index
266 <1> PO_LVI_REG equ 15h ; PCM out Last Valid Index
267 <1> MC_LVI_REG equ 25h ; MIC in Last Valid Index
268 <1>
269 <1> IOC equ BIT31 ; Fire an interrupt whenever this
270 <1> ; buffer is complete.
271 <1> BUP equ BIT30 ; Buffer Underrun Policy.
272 <1>
273 <1> GLOB_CNT_REG equ 2Ch ; Global Control Register
274 <1> GLOB_STS_REG equ 30h ; Global Status register (RO)
275 <1>
276 <1> CTRL_ST_CREADY equ BIT8+BIT9+BIT28 ; Primary Codec Ready
277 <1>
278 <1> CODEC_REG_POWERDOWN equ 26h
279 <1> CODEC_REG_ST equ 26h
280 <1>
281 <1> ; 22/06/2017
282 <1> PO_PICB_REG equ 18h ; PCM Out Position In Current Buffer Register
283 <1>
284 <1> ; 19/11/2023
285 <1> AC97_EA_VRA equ BIT0
286 <1> ; 24/11/2023
287 <1> BCIS equ BIT3 ; Buffer Completion Interrupt Status
288 <1>
289 <1> ;=====
290 <1> ; CODE
291 <1> ;=====
292 <1>
293 <1> ; CODE for INTEL ICH AC'97 AUDIO CONTROLLER
294 <1>
295 <1> DetectICH:
296 <1> ; 22/11/2023
297 <1> ; 19/11/2023
298 <1> ; 01/11/2023 - TRDOS 386 kernel v2.0.7
299 <1> ; 10/06/2017
300 <1> ; 05/06/2017
301 <1> ; 29/05/2017
302 <1> ; 28/05/2017
303 <1> ;mov eax, (ICH_DID << 16) + INTEL_VID
304 <1> ;call pciFindDevice
305 <1> ;jnc short d_ac97_1
306 <1> ;
307 <1> ; 01/11/2023
308 <1> ;mov eax, (NFORCE_DID << 16) + NFORCE_VID
309 <1> ;call pciFindDevice
310 <1> ;jnc short d_ac97_1
311 <1>
312 <1> ; 19/11/2023
313 00013978 BE[9B470100] <1> mov esi, valid_ids ; address of Valid ICH (AC97) Device IDs
314 0001397D B915000000 <1> mov ecx, valid_id_count
315 <1> pfd_1:
316 00013982 AD <1> lodsd
317 00013983 E857000000 <1> call pciFindDevice
318 00013988 730F <1> jnc short d_ac97_1
319 0001398A E2F6 <1> loop pfd_1
320 <1> ;stc
321 <1>

```

	<1>	d_ac97_0:
323	<1>	; couldn't find the audio device!
324	0001398C C3	<1>        retn
325		<1>
326		<1> ; CODE for VIA VT8233 AUDIO CONTROLLER
327		<1>
328		<1> DetectVT8233:
329		<1>        ; 22/11/2023
330		<1>        ; 02/11/2023
331		<1>        ; 01/11/2023 - TRDOS 386 Kernel v2.0.7
332		<1>        ; 06/08/2022 - TRDOS 386 Kernel v2.0.5
333		<1>        ; 10/06/2017
334		<1>        ; 05/06/2017
335		<1>        ; 29/05/2017
336		<1>        ; 03/04/2017
337	0001398D B806115930	<1>        mov     eax, (VT8233_DID << 16) + VIA_VID
338	00013992 E848000000	<1>           call   pciFindDevice
339		<1>        ;        jnc     short d_vt8233_0
340		<1>        ; couldn't find the audio device!
341		<1>        ;        retn
342	00013997 72F3	<1>        ;        jc      short d_ac97_0    ; 28/05/2017
343		<1> d_vt8233_0:
344		<1>        ; 24/03/2017 ('player.asm')
345		<1>        ; 12/11/2016
346		<1>        ; Erdogan Tan - 8/11/2016
347		<1>        ; References: KolibriOS - vt823x.asm (2016)
348		<1>        ;        VIA VT8235 V-Link South Bridge (VT8235-VIA.PDF)(2002)
349		<1>        ;        lowlevel.eu - AC97 (2016)
350		<1>        ;        .wav player for DOS by Jeff Leyda (2002) -this file-
351		<1>        ;        Linux kernel - via82xx.c (2016)
352		<1> d_ac97_1:
353		<1>        ; eax = BUS/DEV/FN
354		<1>        ;        00000000BBBBBBBDDDDFF00000000
355		<1>        ; edx = DEV/VENDOR
356		<1>        ;        DDDDDDDDDDDDDDDVVVVVVVVVVVVVVVVV
357		<1>
358	00013999 A3[D8880100]	<1>        mov     [audio_dev_id], eax
359	0001399E 8915[DC880100]	<1>        mov     [audio_vendor], edx
360		<1>
361		<1>        ; 22/11/2023
362		<1>        ;; init controller
363		<1> ;mov   al, PCI_CMD_REG ; command register (04h)
364		<1> ;call   pciRegRead32
365		<1>        ;
366		<1>        ;; eax = BUS/DEV/FN/REG
367		<1>        ;; edx = STATUS/COMMAND
368		<1>        ;;        SSSSSSSSSSSSSSSCCCCCCCCCCCCCCCC
369		<1>        ;mov   [audio_stats_cmd], edx
370		<1>
371	000139A4 B010	<1>        mov     al, PCI_IO_BASE ; IO base address register (10h)
372		<1> ;mov   al, NAMBAR_REG ; Native Audio Mixer BAR (10h)
373	000139A6 E8BF000000	<1>        call    pciRegRead32
374		<1>
375		<1>        ;cmp   word [audio_vendor], INTEL_VID ; 8086h ; AC'97 ?
376		<1>        ;;jne   short d_vt8233_1
377		<1>        ;;        01/11/2023
378		<1>        ;je     short d_ac97_3
379		<1> ;cmp   word [audio_vendor], NFORCE_VID ; 10DEh ; AC'97
380		<1>        ;jne   short d_vt8233_1
381		<1>        ; 02/11/2023
382	000139AB 66813D[DC880100]06-	<1>        cmp     word [audio_vendor], VIA_VID ; 1106h ; VT8233-VT8237R
382	000139B3 11	<1>
383	000139B4 7411	<1>        je      short d_vt8233_1
384		<1> d_ac97_3:
385		<1>        ;and   dx, 0FFFFeh ; Audio Codec IO_ADDR_MASK
386		<1>        ; 06/08/2022
387	000139B6 80E2FE	<1>        and     dl, 0FEh
388	000139B9 668915[D4880100]	<1>        mov     [NAMBAR], dx
389		<1>
390	000139C0 B014	<1>        mov     al, NABMBAR_REG ; Native Audio Bus Mastering BAR (14h)
391	000139C2 E8A3000000	<1>        call    pciRegRead32
392		<1>
393		<1>        ;;and   dx, 0FFFC0h ; Audio Controller IO_ADDR_MASK
394		<1>        ;; 06/08/2022
395		<1>        ;and   dl, 0C0h
396		<1>        ;mov   [NABMBAR], dx
397		<1>        ;;mov   [audio_io_base], dx
398		<1>        ;
399		<1>        ;jmp   short d_ac97_2
400		<1>        ; 02/11/2023
401		<1>        ; NABMBAR = audio_io_base
402		<1>
403		<1> d_vt8233_1:
404		<1>        ;and   dx, 0FFFC0h ; Audio Controller IO_ADDR_MASK
405		<1>        ; 06/08/2022
406	000139C7 80E2C0	<1>        and     dl, 0C0h
407	000139CA 668915[D6880100]	<1>        mov     [audio_io_base], dx
408		<1>
409		<1> d_ac97_2:
410		<1>        ; 10/06/2017
411	000139D1 B03C	<1>        mov     al, AC97_INT_LINE ; Interrupt Line Register (3ch)
412		<1> ;call   pciRegRead32
413	000139D3 E87F000000	<1>        call    pciRegRead8
414		<1>
415		<1>        ;;and   edx, 0FFh
416		<1>        ; 06/08/2022
417		

```

445         <1> ; the APIC_EN bit should be set to 1.
446         <1> ; The IRQEN must be set to 0 and the PIRQ routed to
447         <1> ; an 8259 interrupt via the IRQ Routing filed (bits[3:0]).
448         <1> ; The corresponding 8259 interrupt must be masked via the
449         <1> ; appropriated bit in the 8259's OCW1 (Interrupt Mask)
450         <1> ; register. The IOAPIC must then be enabled by setting
451         <1> ; the APIC_EN bit in the GEN_CNTL register.
452         <1> ;
453         <1> ;mov     eax, 0F861h ; D31:F0
454         <1> ; ; AL=61h : PIRQ[B] Routing Control Reg, LPC interface
455         <1> ;mov     dl, [audio_intr]
456         <1> ;call    pciRegWrite8
457         <1> ;mov     al, 0D0h ; General Control Register (GEN_CTL)
458         <1> ;call    pciRegRead32
459         <1> ;or      edx, 100h ; Bit 8, APIC_EN (Enable I/O APIC)
460         <1> ;call    pciRegWrite32
461         <1> ;and     edx, ~100h
462         <1> ;call    pciRegWrite32 ; Bit 8, APIC_EN (Disable I/O APIC)
463         <1> ;
464         <1> ;
465         <1> ;mov     dx, 4D1h ; 8259 ELCR2
466         <1> ;in      al, dx
467         <1> ;mov     ah, al
468         <1> ;mov     dx, 4D0h ; 8259 ELCR1
469         <1> ;dec     dl
470         <1> ;in      al, dx
471         <1> ;bts     ax, cx
472         <1> ;mov     dx, 4D0h
473         <1> ;out     dx, al ; set level-triggered mode
474         <1> ;mov     al, ah ; 29/05/2017
475         <1> ;mov     dx, 4D1h
476         <1> ;inc     dl
477         <1> ;out     dx, al ; set level-triggered mode
478         <1> ;
479         <1> ;xor     eax, eax ; 0
480         <1> ;
481         <1> ;retn
482         <1> ;
483         <1> ; CODE for PCI
484         <1> ;
485         <1> pciFindDevice:
486         <1> ; 19/11/2023
487         <1> ; 03/04/2017 ('pci.asm', 20/03/2017)
488         <1> ;
489         <1> ; scan through PCI space looking for a device+vendor ID
490         <1> ;
491         <1> ; Entry: EAX=Device+Vendor ID
492         <1> ;
493         <1> ; Exit: EAX=PCI address if device found
494         <1> ; EDX=Device+Vendor ID
495         <1> ; CY clear if found, set if not found. EAX invalid if CY set.
496         <1> ;
497         <1> ; Destroys: ebx, edi ; 19/11/2023
498         <1> ;
499         <1> ;push    ecx
500         <1> ;push    eax ; * ; 19/11/2023
501         <1> ;push    esi
502         <1> ;push    edi
503         <1> ;
504         <1> ;mov     esi, eax ; save off vend+device ID
505         <1> ; 19/11/2023
506 000139DF 89C3 mov     ebx, eax
507         <1> ;mov     edi, (80000000h - 100h) ; start with bus 0, dev 0 func 0
508 000139E1 BF00000080 mov     edi, 80000000h
509         <1> ;
510         <1> nextPCIddevice:
511         <1> ;add     edi, 100h
512         <1> ;cmp     edi, 80FFF800h ; scanned all devices?
513         <1> ;stc
514         <1> ;je      short PCIScanExit ; not found
515 000139E6 89F8 mov     eax, edi ; read PCI registers
516 000139E8 E87D000000 call    pciRegRead32
517         <1> ; 19/11/2023
518 000139ED 39DA cmp     edx, ebx
519         <1> ;cmp     edx, esi ; found device?
520         <1> ;jne     short nextPCIddevice
521         <1> ;clc
522 000139EF 7412 je      short PCIScanExit ; found
523         <1> ; 19/11/2023
524 000139F1 81FF00F8FF80 cmp     edi, 80FFF800h
525 000139F7 7308 jnb     short pfd_nf ; not found
526 000139F9 81C700010000 add     edi, 100h
527 000139FF EBE5 jmp     short nextPCIddevice
528         <1> pfd_nf:
529 00013A01 F9 stc
530 00013A02 C3 retn
531         <1> PCIScanExit:
532         <1> ;pushf
533 00013A03 B8FFFFFF7F mov     eax, NOT_BIT31 ; 19/03/2017
534 00013A08 21F8 and     eax, edi ; return only bus/dev/fn #
535         <1> ;popf
536         <1> ;
537         <1> ;pop     edi
538         <1> ;pop     esi
539         <1> ;pop     edx ; * ; 19/11/2023
540         <1> ;pop     ecx
541         <1> ;
542 00013A0A C3 retn
543         <1> ;
544         <1> ; 26/11/2023
545         <1> %if 1
546         <1> ;
547         <1> pciRegRead:
548         <1> ; 03/04/2017 ('pci.asm', 20/03/2017)
549         <1> ;
550         <1> ; 8/16/32bit PCI reader
551         <1> ;
552         <1> ; Entry: EAX=PCI Bus/Device/fn/register number
553         <1> ; BIT30 set if 32 bit access requested
554         <1> ; BIT29 set if 16 bit access requested
555         <1> ; otherwise defaults to 8 bit read
556         <1> ;
557         <1> ; Exit: DL,DX,EDX register data depending on requested read size
558         <1> ;
559         <1> ; Note1: this routine is meant to be called via pciRegRead8,
560         <1> ; pciRegRead16 or pciRegRead32, listed below.
561         <1> ;
562         <1> ; Note2: don't attempt to read 32 bits of data from a non dword
563         <1> ; aligned reg number. Likewise, don't do 16 bit reads from
564         <1> ; non word aligned reg #
565         <1> ;
566 00013A0B 53 push    ebx
567 00013A0C 51 push    ecx
568 00013A0D 89C3 mov     ebx, eax ; save eax, dh

```



```

569 00013A0F 88F1      <1>      mov     cl, dh
570                                <1>
571 00013A11 25FFFFFF3F    <1>      and     eax, NOT_PCI32_PCI16    ; clear out data size request
572 00013A16 0D00000080    <1>      or      eax, BIT31              ; make a PCI access request
573 00013A1B 24FC      <1>      and     al, ~3 ; NOT 3          ; force index to be dword
574                                <1>
575 00013A1D 66BAF80C    <1>      mov     dx, PCI_INDEX_PORT
576 00013A21 EF        <1>      out dx, eax          ; write PCI selector
577                                <1>
578 00013A22 66BAFC0C    <1>      mov     dx, PCI_DATA_PORT
579 00013A26 88D8      <1>      mov     al, bl
580 00013A28 2403      <1>      and     al, 3              ; figure out which port to
581 00013A2A 00C2      <1>      add     dl, al          ; read to
582                                <1>
583 00013A2C F7C3000000C0 <1>      test    ebx, PCI32+PCI16
584 00013A32 7507      <1>      jnz     short _pregr0
585 00013A34 EC        <1>      in      al, dx          ; return 8 bits of data
586 00013A35 88C2      <1>      mov     dl, al
587 00013A37 88CE      <1>      mov     dh, cl          ; restore dh for 8 bit read
588 00013A39 EB12      <1>      jmp     short _pregr2
589                                <1>
590 00013A3B F7C300000080 <1>      test    ebx, PCI32
591 00013A41 7507      <1>      jnz     short _pregr1
592 00013A43 66ED      <1>      in      ax, dx
593 00013A45 6689C2    <1>      mov     dx, ax          ; return 16 bits of data
594 00013A48 EB03      <1>      jmp     short _pregr2
595                                <1>
596 00013A4A ED        <1>      in      eax, dx          ; return 32 bits of data
597 00013A4B 89C2      <1>      mov     edx, eax
598                                <1>
599 00013A4D 89D8      <1>      mov     eax, ebx        ; restore eax
600 00013A4F 25FFFFFF3F    <1>      and     eax, NOT_PCI32_PCI16    ; clear out data size request
601 00013A54 59        <1>      pop     ecx
602 00013A55 5B        <1>      pop     ebx
603 00013A56 C3        <1>      retn
604                                <1>
605                                <1>
606 00013A57 25FFFFFF3F    <1>      and     eax, NOT_PCI32_PCI16    ; set up 8 bit read size
607 00013A5C EBAD      <1>      jmp     short pciRegRead ; call generic PCI access
608                                <1>
609                                <1>
610 00013A5E 25FFFFFF3F    <1>      and     eax, NOT_PCI32_PCI16    ; set up 16 bit read size
611 00013A63 0D00000040    <1>      or      eax, PCI16            ; call generic PCI access
612 00013A68 EBA1      <1>      jmp     short pciRegRead
613                                <1>
614                                <1>
615 00013A6A 25FFFFFF3F    <1>      and     eax, NOT_PCI32_PCI16    ; set up 32 bit read size
616 00013A6F 0D00000080    <1>      or      eax, PCI32            ; call generic PCI access
617 00013A74 EB95      <1>      jmp     pciRegRead
618                                <1>
619                                <1>
620                                <1>      ; 03/04/2017 ('pci.asm', 29/11/2016)
621                                <1>      ;
622                                <1>      ; 8/16/32bit PCI writer
623                                <1>      ;
624                                <1>      ; Entry: EAX=PCI Bus/Device/fn/register number
625                                <1>      ;         BIT31 set if 32 bit access requested
626                                <1>      ;         BIT30 set if 16 bit access requested
627                                <1>      ;         otherwise defaults to 8bit read
628                                <1>      ;         DL/DX/EDX data to write depending on size
629                                <1>      ;
630                                <1>      ; Note1: this routine is meant to be called via pciRegWrite8,
631                                <1>      ;         pciRegWrite16 or pciRegWrite32 as detailed below.
632                                <1>      ;
633                                <1>      ; Note2: don't attempt to write 32bits of data from a non dword
634                                <1>      ;         aligned reg number. Likewise, don't do 16 bit writes from
635                                <1>      ;         non word aligned reg #
636                                <1>
637 00013A76 53        <1>      push    ebx
638 00013A77 51        <1>      push    ecx
639 00013A78 89C3      <1>      mov     ebx, eax          ; save eax, edx
640 00013A7A 89D1      <1>      mov     ecx, edx
641 00013A7C 25FFFFFF3F    <1>      and     eax, NOT_PCI32_PCI16    ; clear out data size request
642 00013A81 0D00000080    <1>      or      eax, BIT31              ; make a PCI access request
643 00013A86 24FC      <1>      and     al, ~3 ; NOT 3          ; force index to be dword
644                                <1>
645 00013A88 66BAF80C    <1>      mov     dx, PCI_INDEX_PORT
646 00013A8C EF        <1>      out dx, eax          ; write PCI selector
647                                <1>
648 00013A8D 66BAFC0C    <1>      mov     dx, PCI_DATA_PORT
649 00013A91 88D8      <1>      mov     al, bl
650 00013A93 2403      <1>      and     al, 3              ; figure out which port to
651 00013A95 00C2      <1>      add     dl, al          ; write to
652                                <1>
653 00013A97 F7C3000000C0 <1>      test    ebx, PCI32+PCI16
654 00013A9D 7505      <1>      jnz     short _pregw0
655 00013A9F 88C8      <1>      mov     al, cl          ; put data into al
656 00013AA1 EE        <1>      out     dx, al
657 00013AA2 EB12      <1>      jmp     short _pregw2
658                                <1>
659 00013AA4 F7C300000080 <1>      test    ebx, PCI32
660 00013AAA 7507      <1>      jnz     short _pregw1
661 00013AAC 6689C8    <1>      mov     ax, cx          ; put data into ax
662 00013AAF 66EF      <1>      out     dx, ax
663 00013AB1 EB03      <1>      jmp     short _pregw2
664                                <1>
665 00013AB3 89C8      <1>      mov     eax, ecx          ; put data into eax
666 00013AB5 EF        <1>      out     dx, eax
667                                <1>
668 00013AB6 89D8      <1>      mov     eax, ebx        ; restore eax
669 00013AB8 25FFFFFF3F    <1>      and     eax, NOT_PCI32_PCI16    ; clear out data size request
670 00013ABD 89CA      <1>      mov     edx, ecx        ; restore dx
671 00013ABF 59        <1>      pop     ecx
672 00013AC0 5B        <1>      pop     ebx
673 00013AC1 C3        <1>      retn
674                                <1>
675                                <1>
676 00013AC2 25FFFFFF3F    <1>      and     eax, NOT_PCI32_PCI16    ; set up 8 bit write size
677 00013AC7 EBAD      <1>      jmp     short pciRegWrite ; call generic PCI access
678                                <1>
679                                <1>
680 00013AC9 25FFFFFF3F    <1>      and     eax, NOT_PCI32_PCI16    ; set up 16 bit write size
681 00013ACE 0D00000040    <1>      or      eax, PCI16            ; call generic PCI access
682 00013AD3 EBA1      <1>      jmp     short pciRegWrite
683                                <1>
684                                <1>
685 00013AD5 25FFFFFF3F    <1>      and     eax, NOT_PCI32_PCI16    ; set up 32 bit write size
686 00013ADA 0D00000080    <1>      or      eax, PCI32            ; call generic PCI access
687 00013ADF EB95      <1>      jmp     pciRegWrite
688                                <1>
689                                <1>      %endif
690                                <1>
691                                <1>      ; 26/11/2023 - temporary
692                                <1>      %if 0

```

```

693 <1>
694 <1> ; PLAYWAV3.COM, ac97_vra.asm, 19/11/2023, Erdogan Tan
695 <1>
696 <1> ;=====
697 <1> ; 8/16/32bit PCI reader
698 <1>
699 <1> ; Entry: EAX=PCI Bus/Device/fn/register number
700 <1> ;         BIT30 set if 32 bit access requested
701 <1> ;         BIT29 set if 16 bit access requested
702 <1> ;         otherwise defaults to 8 bit read
703 <1>
704 <1> ; Exit: DL,DX,EDX register data depending on requested read size
705 <1>
706 <1> ; Note: this routine is meant to be called via pciRegRead8, pciRegread16,
707 <1> ;       or pciRegRead32, listed below.
708 <1>
709 <1> ; Note2: don't attempt to read 32bits of data from a non dword aligned reg
710 <1> ;       number. Likewise, don't do 16bit reads from non word aligned reg #
711 <1>
712 <1> pciRegRead:
713 <1>     push    ebx
714 <1>     push    ecx
715 <1>     mov     ebx, eax                ; save eax, dh
716 <1>     mov     cl, dh
717 <1>     and     eax, (~PCI32)+PCI16    ; clear out data size request
718 <1>     or      eax, BIT31             ; make a PCI access request
719 <1>     and     al, ~3 ; NOT 3        ; force index to be dword
720 <1>
721 <1>     mov     dx, PCI_INDEX_PORT
722 <1>     out     dx, eax                ; write PCI selector
723 <1>
724 <1>     mov     dx, PCI_DATA_PORT
725 <1>     mov     al, bl
726 <1>     and     al, 3
727 <1>     add     dl, al                ; figure out which port to
728 <1>                                     ; read to
729 <1>
730 <1>     in      eax, dx                ; do 32bit read
731 <1>     test    ebx, PCI32
732 <1>     jz      short _pregr1
733 <1>     mov     edx, eax                ; return 32bits of data
734 <1>
735 <1> _pregr1:
736 <1>     mov     dx, ax                ; return 16bits of data
737 <1>     test    ebx, PCI32+PCI16
738 <1>     jnz     short _pregr2
739 <1>     mov     dh, cl                ; restore dh for 8 bit read
740 <1>
741 <1> _pregr2:
742 <1>     mov     eax, ebx                ; restore eax
743 <1>     and     eax, (~PCI32)+PCI16    ; clear out data size request
744 <1>     pop     ecx
745 <1>     pop     ebx
746 <1>     retn
747 <1>
748 <1> pciRegRead8:
749 <1>     and     eax, (~PCI16)+PCI32    ; set up 8 bit read size
750 <1>     jmp     short pciRegRead      ; call generic PCI access
751 <1>
752 <1> pciRegRead16:
753 <1>     and     eax, (~PCI16)+PCI32    ; set up 16 bit read size
754 <1>     or      eax, PCI16             ; call generic PCI access
755 <1>     jmp     short pciRegRead
756 <1>
757 <1> pciRegRead32:
758 <1>     and     eax, (~PCI16)+PCI32    ; set up 32 bit read size
759 <1>     or      eax, PCI32             ; call generic PCI access
760 <1>     jmp     short pciRegRead
761 <1>
762 <1> ;=====
763 <1> ; 8/16/32bit PCI writer
764 <1>
765 <1> ; Entry: EAX=PCI Bus/Device/fn/register number
766 <1> ;         BIT31 set if 32 bit access requested
767 <1> ;         BIT30 set if 16 bit access requested
768 <1> ;         otherwise defaults to 8bit read
769 <1> ;         DL/DX/EDX data to write depending on size
770 <1>
771 <1> ; note: this routine is meant to be called via pciRegWrite8, pciRegwrite16,
772 <1> ;       or pciRegWrite32 as detailed below.
773 <1>
774 <1> ; Note2: don't attempt to write 32bits of data from a non dword aligned reg
775 <1> ;       number. Likewise, don't do 16bit writes from non word aligned reg #
776 <1>
777 <1> pciRegWrite:
778 <1>     push    ebx
779 <1>     push    ecx
780 <1>     mov     ebx, eax                ; save eax, dx
781 <1>     mov     cx, dx
782 <1>     or      eax, BIT31             ; make a PCI access request
783 <1>     and     eax, ~PCI16 ; NOT PCI16 ; clear out data size request
784 <1>     and     al, ~3 ; NOT 3        ; force index to be dword
785 <1>
786 <1>     mov     dx, PCI_INDEX_PORT
787 <1>     out     dx, eax                ; write PCI selector
788 <1>
789 <1>     mov     dx, PCI_DATA_PORT
790 <1>     mov     al, bl
791 <1>     and     al, 3
792 <1>     add     dl, al                ; figure out which port to
793 <1>                                     ; write to
794 <1>
795 <1>     mov     eax, edx                ; put data into eax
796 <1>     mov     ax, cx
797 <1>
798 <1>     out     dx, al
799 <1>     test    ebx, PCI16+PCI32
800 <1>     jz      short _pregw1          ; only 8bit access? bail
801 <1>
802 <1>     out     dx, ax
803 <1>     test    ebx, PCI16
804 <1>     jnz     short _pregw1          ; 16bit requested? bail
805 <1>
806 <1>     out     dx, eax                ; write full 32bit
807 <1>
808 <1> _pregw1:
809 <1>     mov     eax, ebx                ; restore eax
810 <1>     and     eax, (~PCI32)+PCI16    ; clear out data size request
811 <1>     mov     dx, cx                ; restore dx
812 <1>     pop     ecx
813 <1>     pop     ebx
814 <1>     ret
815 <1>
816 <1> pciRegWrite8:
817 <1>     and     eax, (~PCI16)+PCI32    ; set up 8 bit write size
818 <1>     jmp     short pciRegWrite     ; call generic PCI access

```

```

817 <1> pciRegWrite16:
818 <1>     and     eax, (~PCI16)+PCI32          ; set up 16 bit write size
819 <1>     or      eax, PCI16                    ; call generic PCI access
820 <1>     jmp     short pciRegWrite
821 <1>
822 <1> pciRegWrite32:
823 <1>     and     eax, (~PCI16)+PCI32          ; set up 32 bit write size
824 <1>     or      eax, PCI32                    ; call generic PCI access
825 <1>     jmp     short pciRegWrite
826 <1>
827 <1> ;=====
828 <1>
829 <1> %endif
830 <1>
831 <1> init_codec:
832 <1>     ; 06/08/2022 - TRDOS 386 v2.0.5
833 <1>     ; 05/06/2017
834 <1>     ; 28/05/2017 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
835 <1>     ;
836 00013AE1 A1[D8880100] <1>     mov     eax, [audio_dev_id]
837 00013AE6 B041 <1>     mov     al, VIA_ACLINK_CTRL
838 00013AE8 E86AFFFFFF <1>     call    pciRegRead8
839 <1>     ; ?
840 00013AED B040 <1>     mov     al, VIA_ACLINK_STAT
841 00013AEF E863FFFFFF <1>     call    pciRegRead8
842 00013AF4 F6C201 <1>     test    dl, VIA_ACLINK_C00_READY
843 00013AF7 7508 <1>     jnz     short _codec_ready_1
844 00013AF9 E80D000000 <1>     call    reset_codec
845 00013AFE 7305 <1>     jnc     short _codec_ready_2 ; eax = 1
846 00013B00 C3 <1>     retn
847 <1> _codec_ready_1:
848 <1>     ;mov     eax, 1
849 <1>     ; 06/08/2022
850 00013B01 29C0 <1>     sub     eax, eax
851 00013B03 FEC0 <1>     inc     al
852 <1>     ; eax = 1
853 <1> _codec_ready_2:
854 00013B05 E880000000 <1>     call    codec_io_w16
855 <1> detect_codec:
856 00013B0A C3 <1>     retn
857 <1>
858 <1> reset_codec:
859 <1>     ; 16/04/2017
860 <1>     ; 23/03/2017
861 <1>     ; ('codec.asm')
862 <1>     ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
863 00013B0B A1[D8880100] <1>     mov     eax, [audio_dev_id]
864 00013B10 B041 <1>     mov     al, VIA_ACLINK_CTRL
865 00013B12 B2E0 <1>     mov     dl, VIA_ACLINK_CTRL_ENABLE + VIA_ACLINK_CTRL_RESET + VIA_ACLINK_CTRL_SYNC
866 00013B14 E8A9FFFFFF <1>     call    pciRegWrite8
867 <1>
868 00013B19 E843000000 <1>     call    delay_100ms ; wait 100 ms
869 <1> _rc_cold:
870 00013B1E E811000000 <1>     call    cold_reset
871 00013B23 7301 <1>     jnc     short _reset_codec_ok
872 <1>
873 <1>     ; 16/04/2017
874 <1>     ;xor     eax, eax ; timeout error
875 <1>     ;stc
876 00013B25 C3 <1>     retn
877 <1>
878 <1> _reset_codec_ok:
879 <1>     ; 01/09/2020
880 <1>     ; 15/08/2020
881 <1>     ; 27/07/2020
882 <1>     ; also reset codec by using index control register 0 of AD1980 or ALC655
883 <1>     ; (to fix line out -2 channels audio playing- problem on AD1980 codec)
884 <1>
885 00013B26 29C0 <1>     sub     eax, eax
886 <1>     ; 02/11/2023
887 <1>     ;mov     edx, CODEC_RESET_REG ; 00h ; Reset register
888 00013B28 31D2 <1>     xor     edx, edx ; 00h ; Reset register
889 00013B2A E8BE000000 <1>     call    codec_write
890 <1>
891 <1>     ;sub     eax, eax
892 <1>     ; 01/09/2020
893 <1>     ; 15/08/2020
894 <1>     ; AD1980 BugFix
895 <1>     ; (set HPSEL -headphone amp to be driven from mixer- and
896 <1>     ; CLDIS -center and LFE disable- bits)
897 <1>     ;mov     eax, 0C00h ; HPSEL = bit 10, CLDIS = bit 11 ; 01/09/2020
898 <1>     ;mov     edx, CODEC_MISC_CTRL_BITS_REG ; 76h ; Misc Ctrl Bits ; AD1980
899 <1>     ;call    codec_write
900 <1>
901 00013B2F 31C0 <1>     xor     eax, eax
902 <1>     ;mov     al, VIA_ACLINK_C00_READY ; 1
903 00013B31 FEC0 <1>     inc     al
904 00013B33 C3 <1>     retn
905 <1>
906 <1> cold_reset:
907 <1>     ; 06/08/2022 - TRDOS 386 v2.0.5
908 <1>     ; 16/04/2017
909 <1>     ; 23/03/2017
910 <1>     ; ('codec.asm')
911 <1>     ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
912 <1>     ;mov     eax, [audio_dev_id]
913 <1>     ;mov     al, VIA_ACLINK_CTRL
914 00013B34 30D2 <1>     xor     dl, dl ; 0
915 00013B36 E887FFFFFF <1>     call    pciRegWrite8
916 <1>
917 00013B3B E821000000 <1>     call    delay_100ms ; wait 100 ms
918 <1>
919 <1>     ;; ACLink on, deassert ACLink reset, VSR, SGD data out
920 <1>     ;; note - FM data out has trouble with non VRA codecs !!
921 <1>
922 <1>     ;mov     eax, [audio_dev_id]
923 <1>     ;mov     al, VIA_ACLINK_CTRL
924 00013B40 B2CC <1>     mov     dl, VIA_ACLINK_CTRL_INIT
925 00013B42 E87BFFFFFF <1>     call    pciRegWrite8
926 <1>
927 <1>     ;mov     ecx, 16 ; total 2s
928 <1>     ; 06/08/2022
929 <1>     ;sub     ecx, ecx
930 <1>     ; 22/11/2023
931 <1>     ; ecx = 0 (from 'delay_100ms')
932 00013B47 B110 <1>     mov     cl, 16
933 <1> _crst_wait:
934 <1>     ;mov     eax, [audio_dev_id]
935 00013B49 B040 <1>     mov     al, VIA_ACLINK_STAT
936 00013B4B E807FFFFFF <1>     call    pciRegRead8
937 <1>
938 00013B50 F6C201 <1>     test    dl, VIA_ACLINK_C00_READY
939 00013B53 750B <1>     jnz     short _crst_ok
940 <1>

```

```

941 00013B55 51      <1>      push    ecx
942 00013B56 E806000000 <1>      call   delay_100ms
943 00013B5B 59      <1>      pop     ecx
944                <1>
945 00013B5C 49      <1>      dec     ecx
946 00013B5D 75EA    <1>      jnz     short _crst_wait
947                <1>
948                <1> _crst_fail:
949                <1>      stc
950                <1> _crst_ok:
951 00013B60 C3      <1>      retn
952                <1>
953                <1> delay_100ms:
954                <1>      ; 29/05/2017
955                <1>      ; 24/03/2017 ('codec.asm')
956                <1>      ; wait 100 ms
957 00013B61 B990010000 <1>      mov     ecx, 400 ; 400*0.25ms
958                <1> _delay_x_ms:
959 00013B66 E803000000 <1>      call   delay1_4ms
960 00013B6B E2F9    <1>      loop   _delay_x_ms
961 00013B6D C3      <1>      retn
962                <1>
963                <1> ; delay1_4ms - Delay for 1/4 millisecond.
964                <1> ;
965                <1> ; 1ms = 1000us
966                <1> ; Entry:
967                <1> ; None
968                <1> ; Exit:
969                <1> ; None
970                <1> ;
971                <1> ; Modified:
972                <1> ; None
973                <1> ;
974                <1> ; 29/05/2017
975                <1> ; 23/04/2017
976                <1> ; 05/03/2017 (TRDOS 386)
977                <1> ; ('UTILS.ASM')
978 00013B6E 50      <1>      delay1_4ms:
979 00013B6F 51      <1>      push    eax
980 00013B70 B110    <1>      push    ecx
981                <1>      mov     cl, 16 ; close enough.
982 00013B72 E461    <1>      in      al, PORTB ; 61h
983                <1>
984 00013B74 2410    <1>      and     al, REFRESH_STATUS ; 10h
985 00013B76 88C5    <1>      mov     ch, al ; Start toggle state
986                <1> _d4ms1:
987 00013B78 E461    <1>      in      al, PORTB ; Read system control port
988                <1>
989 00013B7A 2410    <1>      and     al, REFRESH_STATUS ; Refresh toggles 15.085 microseconds
990 00013B7C 38C5    <1>      cmp     ch, al
991 00013B7E 74F8    <1>      je      short _d4ms1 ; wait for state change
992                <1>
993 00013B80 88C5    <1>      mov     ch, al ; Update with new state
994 00013B82 FEC9    <1>      dec     cl
995 00013B84 75F2    <1>      jnz     short _d4ms1
996                <1>
997 00013B86 F8      <1>      cld ; 29/05/2017
998                <1>
999                <1>      pop     ecx
1000                <1>      pop     eax
1001 00013B89 C3      <1>      retn
1002                <1>
1003                <1> ; 10/04/2017 (TRDOS 386)
1004                <1> ; 12/11/2016
1005                <1>
1006                <1> codec_io_w16: ;w32
1007                <1> ; ('codec.asm')
1008 00013B8A 668B15[D6880100] <1>      mov     dx, [audio_io_base]
1009 00013B91 6681C28000 <1>      add     dx, VIA_REG_AC97
1010 00013B96 EF      <1>      out     dx, eax
1011 00013B97 C3      <1>      retn
1012                <1>
1013                <1> codec_io_r16: ;r32
1014                <1> ; ('codec.asm')
1015 00013B98 668B15[D6880100] <1>      mov     dx, [audio_io_base]
1016 00013B9F 6681C28000 <1>      add     dx, VIA_REG_AC97
1017 00013BA4 ED      <1>      in      eax, dx
1018 00013BA5 C3      <1>      retn
1019                <1>
1020                <1> ctrl_io_w8:
1021                <1> ; ('codec.asm')
1022 00013BA6 660315[D6880100] <1>      add     dx, [audio_io_base]
1023 00013BAD EE      <1>      out     dx, al
1024 00013BAE C3      <1>      retn
1025                <1>
1026                <1> ctrl_io_r8:
1027                <1> ; ('codec.asm')
1028 00013BAF 660315[D6880100] <1>      add     dx, [audio_io_base]
1029 00013BB6 EC      <1>      in      al, dx
1030 00013BB7 C3      <1>      retn
1031                <1>
1032                <1> ctrl_io_w32:
1033                <1> ; ('codec.asm')
1034 00013BB8 660315[D6880100] <1>      add     dx, [audio_io_base]
1035 00013BBF EF      <1>      out     dx, eax
1036 00013BC0 C3      <1>      retn
1037                <1>
1038                <1> ctrl_io_r32:
1039                <1> ; ('codec.asm')
1040 00013BC1 660315[D6880100] <1>      add     dx, [audio_io_base]
1041 00013BC8 ED      <1>      in      eax, dx
1042                <1> _cr_not_rdy: ; 06/08/2022
1043 00013BC9 C3      <1>      retn
1044                <1>
1045                <1> codec_read:
1046                <1> ; 06/08/2022 - TRDOS 386 v2.0.5
1047                <1> ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
1048                <1> ; Use only primary codec.
1049                <1> ; eax = register
1050 00013BCA C1E010 <1>      shl     eax, VIA_REG_AC97_CMD_SHIFT
1051 00013BCD 0D00008002 <1>      or      eax, VIA_REG_AC97_PRIMARY_VALID + VIA_REG_AC97_READ
1052                <1>
1053 00013BD2 E8B3FFFFFF <1>      call    codec_io_w16
1054                <1>
1055                <1> ; codec_valid
1056 00013BD7 E825000000 <1>      call    codec_check_ready
1057                <1> ;jnc short _cr_ok
1058                <1> ;retn
1059                <1> ; 06/08/2022
1060 00013BDC 72EB    <1>      jc      short _cr_not_rdy
1061                <1> ; ecx <= 20
1062                <1> _cr_ok:
1063                <1> ; wait 25 ms
1064                <1> ;mov     ecx, 80 ; (100*0.25 ms)

```

```

1065 <1> ; 06/08/2022
1066 <1> ;xor ecx, ecx
1067 00013BDE B150 <1> mov cl, 80
1068 <1> ; ecx = 80
1069 <1> _cr_wloop:
1070 00013BE0 E889FFFFFF <1> call delay1_4ms
1071 00013BE5 E2F9 <1> loop _cr_wloop
1072 <1>
1073 00013BE7 E8ACFFFFFF <1> call codec_io_r16
1074 <1> ; 06/08/2022
1075 <1> ;and eax, 0FFFFh
1076 00013BEC C3 <1> retn
1077 <1>
1078 <1> codec_write:
1079 <1> ; 06/08/2022 - TRDOS 386 v2.0.5
1080 <1> ; 12/11/2016 - Erdogan Tan (Ref: kolibriOS, vt823x.asm)
1081 <1> ; Use only primary codec.
1082 <1>
1083 <1> ; eax = data (volume)
1084 <1> ; edx = register (mixer register)
1085 <1>
1086 00013BED C1E210 <1> shl edx, VIA_REG_AC97_CMD_SHIFT
1087 <1>
1088 <1> ; 02/11/2023 (shl eax, 0)
1089 <1> ;shl eax, VIA_REG_AC97_DATA_SHIFT ; shl eax, 0
1090 00013BF0 09C2 <1> or edx, eax
1091 <1>
1092 00013BF2 B800000000 <1> mov eax, VIA_REG_AC97_CODEC_ID_PRIMARY
1093 00013BF7 C1E01E <1> shl eax, VIA_REG_AC97_CODEC_ID_SHIFT
1094 00013BFA 09D0 <1> or eax, edx
1095 <1>
1096 00013BFC E889FFFFFF <1> call codec_io_w16
1097 <1> ;mov [codec.regs+esi], ax
1098 <1>
1099 <1> ;call codec_check_ready
1100 <1> ;retn
1101 <1> ;jmp short _codec_check_ready
1102 <1>
1103 <1> codec_check_ready:
1104 <1> ; 06/08/2022 - TRDOS 386 v2.0.5
1105 <1> ; 12/11/2016 - Erdogan Tan (Ref: kolibriOS, vt823x.asm)
1106 <1>
1107 <1> _codec_check_ready:
1108 <1> ;mov ecx, 20; total 2s
1109 <1> ; 06/08/2022
1110 00013C01 29C9 <1> sub ecx, ecx
1111 <1> ;mov cl, 20
1112 <1> ; 02/11/2023
1113 00013C03 B10A <1> mov cl, 10 ; wait 1s
1114 <1> _ccr_wait:
1115 00013C05 51 <1> push ecx
1116 <1>
1117 00013C06 E88DFFFFFF <1> call codec_io_r16
1118 00013C08 A900000001 <1> test eax, VIA_REG_AC97_BUSY
1119 00013C10 740B <1> jz short _ccr_ok
1120 <1>
1121 00013C12 E84AFFFFFF <1> call delay_100ms
1122 <1>
1123 00013C17 59 <1> pop ecx
1124 <1>
1125 00013C18 49 <1> dec ecx
1126 00013C19 75EA <1> jnz short _ccr_wait
1127 <1>
1128 00013C1B F9 <1> stc
1129 00013C1C C3 <1> retn
1130 <1>
1131 <1> _ccr_ok:
1132 00013C1D 59 <1> pop ecx
1133 00013C1E 25FFFF0000 <1> and eax, 0FFFFh
1134 00013C23 C3 <1> retn
1135 <1>
1136 <1> codec_config:
1137 <1> ; 02/11/2023 - TRDOS 386 kernel v2.0.7
1138 <1> ; 06/08/2022 - TRDOS 386 kernel v2.0.5
1139 <1> ; 10/06/2017
1140 <1> ; 29/05/2017
1141 <1> ; 24/04/2017
1142 <1> ; 21/04/2017
1143 <1> ; 16/04/2017 (TRDOS 386 kernel)
1144 <1> ; 15/11/2016 ('codec.asm', 'player.com')
1145 <1> ; 14/11/2016
1146 <1> ; 12/11/2016 - Erdogan Tan
1147 <1> ; (Ref: kolibriOS, 'setup_codec', codec.inc)
1148 <1>
1149 00013C24 B802020000 <1> mov eax, 0202h
1150 00013C29 66A3[0C890100] <1> mov [audio_master_volume], ax
1151 <1>
1152 <1> ;mov ax, 1F1Fh ; 31,31
1153 <1> ; 02/11/2023
1154 00013C2F 66B80B0B <1> mov ax, 0B0Bh
1155 <1> ;mov edx, CODEC_MASTER_VOL_REG ; 02h ; Line Out
1156 <1> ; 06/08/2022
1157 00013C33 29D2 <1> sub edx, edx
1158 00013C35 B202 <1> mov dl, CODEC_MASTER_VOL_REG ; 02h ; Line Out
1159 00013C37 E8B1FFFFFF <1> call codec_write
1160 <1> ;jc short cconfig_error
1161 <1>
1162 <1> ;mov eax, 0202h
1163 00013C3C 66B80202 <1> mov ax, 0202h
1164 <1> ;mov edx, CODEC_PCM_OUT_REG ; 18h ; wave Output (Stereo)
1165 <1> ; 06/08/2022
1166 00013C40 29D2 <1> sub edx, edx
1167 00013C42 B218 <1> mov dl, CODEC_PCM_OUT_REG ; 18h ; wave Output (Stereo)
1168 00013C44 E8A4FFFFFF <1> call codec_write
1169 <1> ;jc short cconfig_error
1170 <1>
1171 <1> ;mov eax, 0202h
1172 00013C49 66B80202 <1> mov ax, 0202h
1173 <1> ;mov edx, CODEC_AUX_VOL ; 04h ; CODEC_HP_VOL_REG ; HeadPhone
1174 <1> ; 06/08/2022
1175 00013C4D 29D2 <1> sub edx, edx
1176 00013C4F B204 <1> mov dl, CODEC_AUX_VOL ; 04h ; CODEC_HP_VOL_REG ; HeadPhone
1177 00013C51 E897FFFFFF <1> call codec_write
1178 <1> ;jc short cconfig_error
1179 <1>
1180 <1> ;mov eax, 08h
1181 <1> ;mov ax, 08h
1182 00013C56 66B80880 <1> mov ax, 8008h ; Mute
1183 <1> ;mov edx, 0Ch ; AC97_PHONE_VOL ; TAD Input (Mono)
1184 <1> ; 06/08/2022
1185 00013C5A 29D2 <1> sub edx, edx
1186 00013C5C B20C <1> mov dl, 0Ch ; AC97_PHONE_VOL ; TAD Input (Mono)
1187 00013C5E E88AFFFFFF <1> call codec_write
1188 <1> ;jc short cconfig_error

```

```

1189 <1>
1190 <1> ;mov     eax, 0808h
1191 00013C63 66B80808 <1> mov     ax, 0808h
1192 <1> ;mov     edx, CODEC_LINE_IN_VOL_REG ; 10h ; Line Input (Stereo)
1193 <1> ; 06/08/2022
1194 00013C67 29D2 <1> sub     edx, edx
1195 00013C69 B210 <1> mov     dl, CODEC_LINE_IN_VOL_REG ; 10h ; Line Input (Stereo)
1196 00013C6B E87DFFFFFF <1> call    codec_write
1197 <1> ;jc      short cconfig_error
1198 <1>
1199 <1> ;mov     eax, 0808h
1200 00013C70 66B80808 <1> mov     ax, 0808h
1201 <1> ;mov     edx, CODEC_CD_VOL_REG ; 12h ; CR Input (Stereo)
1202 <1> ; 06/08/2022
1203 00013C74 29D2 <1> sub     edx, edx ; 02/11/2023
1204 00013C76 B212 <1> mov     dl, CODEC_CD_VOL_REG ; 12h ; CR Input (Stereo)
1205 00013C78 E870FFFFFF <1> call    codec_write
1206 <1> ;jc      short cconfig_error
1207 <1>
1208 <1> ;mov     eax, 0808h
1209 00013C7D 66B80808 <1> mov     ax, 0808h
1210 <1> ;mov     edx, CODEC_AUX_VOL_REG ; 16h ; Aux Input (Stereo)
1211 <1> ; 06/08/2022
1212 00013C81 29D2 <1> sub     edx, edx
1213 00013C83 B216 <1> mov     dl, CODEC_AUX_VOL_REG ; 16h ; Aux Input (Stereo)
1214 <1> ;call    codec_write
1215 <1> ;jc      short cconfig_error
1216 00013C85 E963FFFFFF <1> jmp     codec_write ; 10/06/2017
1217 <1>
1218 <1> ; ; Extended Audio Status (2Ah)
1219 <1> ; mov     eax, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
1220 <1> ; call    codec_read
1221 <1> ; and     eax, 0FFFFh - 2 ; clear DRA (BIT1)
1222 <1> ; ; or     eax, 1 ; set VRA (BIT0)
1223 <1> ; or     eax, 5 ; VRA (BIT0) & S/PDIF (BIT2) ; 14/11/2016
1224 <1> ; mov     edx, CODEC_EXT_AUDIO_CTRL_REG
1225 <1> ; call    codec_write
1226 <1> ;jc      short cconfig_error
1227 <1>
1228 <1> ;set_sample_rate:
1229 <1> ;movzx   eax, word [audio_freq]
1230 <1> ; mov     ax, [audio_freq]
1231 <1> ; mov     edx, CODEC_PCM_FRONT_DACRATE_REG ; 2Ch ; PCM Front DAC Rate
1232 <1> ; call    codec_write
1233 <1> ; ; retn
1234 <1> ; jmp     codec_write
1235 <1>
1236 <1> ;cconfig_error:
1237 <1> ; retn
1238 <1>
1239 <1> vt8233_int_handler:
1240 <1> ; 05/06/2024
1241 <1> ; 04/06/2024 - TRDOS 386 v2.0.8
1242 <1> ; 27/07/2020
1243 <1> ; 22/07/2020
1244 <1> ; Interrupt Handler for VIA VT8237R Audio Controller
1245 <1> ; Note: called by 'dev_IRQ_service'
1246 <1> ; 14/10/2017
1247 <1> ; 09/10/2017, 10/10/2017, 12/10/2017
1248 <1> ; 13/06/2017
1249 <1> ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
1250 <1> ; 24/03/2017 - 'PLAYER.COM' ('player.asm')
1251 <1>
1252 <1> ;push     eax ; * must be saved !
1253 <1> ;push     edx
1254 <1> ;push     ecx
1255 <1> ;push     ebx ; * must be saved !
1256 <1> ;push     esi
1257 <1> ;push     edi
1258 <1>
1259 <1> ;cmp      byte [audio_busy], 1
1260 <1> ;jnb      short _ih0 ; 09/10/2017
1261 <1>
1262 <1> ;mov      byte [audio_flag_eol], 0
1263 <1>
1264 00013C8A 66BA0000 <1> mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
1265 00013C8E E81CFFFFFF <1> call    ctrl_io_r8
1266 <1>
1267 00013C93 A880 <1> test    al, VIA_REG_STAT_ACTIVE
1268 00013C95 7417 <1> jz      short _ih0 ; 09/10/2017
1269 <1>
1270 00013C97 2407 <1> and     al, VIA_REG_STAT_EOL + VIA_REG_STAT_FLAG + VIA_REG_STAT_STOPPED
1271 00013C99 A2[0B890100] <1> mov     [audio_flag_eol], al
1272 00013C9E 740E <1> jz      short _ih0 ; 09/10/2017
1273 <1>
1274 <1> ; 09/10/2017
1275 <1> ;mov      byte [audio_busy], 1
1276 <1>
1277 00013CA0 803D[0A890100]01 <1> cmp     byte [audio_play_cmd], 1
1278 00013CA7 7315 <1> jnb     short _ih1 ; 10/10/2017
1279 <1>
1280 00013CA9 E848000000 <1> call    channel_reset
1281 <1> _ih0:
1282 <1> ; 09/10/2017
1283 00013CAE A0[0B890100] <1> mov     al, [audio_flag_eol] ;; ack ;;
1284 00013CB3 66BA0000 <1> mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
1285 00013CB7 E8EAFEFFFF <1> call    ctrl_io_w8
1286 00013CBC EB37 <1> jmp     short _ih4
1287 <1> _ih1:
1288 <1> vt8233_tuneLoop:
1289 00013CBE A0[0B890100] <1> mov     al, [audio_flag_eol] ;; ack ;;
1290 00013CC3 66BA0000 <1> mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STATUS
1291 00013CC7 E8DAFEFFFF <1> call    ctrl_io_w8
1292 <1>
1293 <1> ; 22/07/2020
1294 <1> ; 12/10/2017
1295 <1> ;mov      byte [audio_flag], 0 ; Reset
1296 <1>
1297 <1> ; 10/10/2017
1298 <1> ; 09/10/2017
1299 <1> ;test     byte [audio_flag_eol], VIA_REG_STAT_FLAG
1300 <1> ;jz      short _ih2 ; EOL
1301 <1>
1302 <1> ; 22/07/2020
1303 <1> ; 14/10/2017
1304 <1> ;test     byte [audio_flag_eol], VIA_REG_STAT_EOL
1305 <1> ;jnz     short _ih2 ; EOL
1306 <1> ; ; (Half Buffer 2 has been completed
1307 <1> ; ; and Half Buffer 1 will be played.)
1308 <1>
1309 <1> ; FLAG
1310 <1> ; (Half Buffer 1 has been completed
1311 <1> ; and Half Buffer 2 will be played.)
1312 <1>

```

```

1313 <1> ; 14/10/2017
1314 <1> ;; (Continue to play.)
1315 <1> ;mov al, VIA_REG_CTRL_INT
1316 <1> ;or al, VIA_REG_CTRL_START
1317 <1> ;mov dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
1318 <1> ;call ctrl_io_w8
1319 <1> ; 12/10/2017
1320 <1> ;mov byte [audio_flag], 1
1321 <1>
1322 <1> ; 22/07/2020
1323 <1> ;inc byte [audio_flag] ; = 1
1324 <1> _ih2:
1325 <1> ; 10/10/2017
1326 00013CCC 8B3D[F0880100] <1> mov edi, [audio_dma_buff]
1327 <1>
1328 <1> ; 05/06/2024
1329 <1> ; 04/06/2024
1330 <1> ;mov ecx, [audio_dmabuff_size]
1331 <1> ;shr ecx, 1 ; dma buff size / 2 = half buffer size
1332 <1> ;mov ecx, [audio_buff_size]
1333 <1> ;and cl, ~1 ; word aligned
1334 <1> ; 05/06/2024
1335 00013CD2 8B0D[F8880100] <1> mov ecx, [dma_hbuff_size] ; half buffer size
1336 <1>
1337 <1> ; 22/07/2020
1338 <1> ; 12/10/2017
1339 <1> ;cmp byte [audio_flag], 0
1340 <1> ;ja short _ih3 ; Playing Half Buffer 2 (Current: FLAG)
1341 <1>
1342 <1> ; 27/07/2020
1343 <1> ; 22/07/2020
1344 00013CD8 F605[FC880100]01 <1> test byte [audio_flag], 1 ; Current flag value
1345 00013CDF 7402 <1> jz short _ih3 ; Half Buffer 1 must be filled
1346 <1>
1347 <1> ; Half Buffer 2 must be filled
1348 00013CE1 01CF <1> add edi, ecx
1349 <1> _ih3:
1350 <1> ; Update half buffer 2 while playing half buffer 1
1351 <1> ; Update half buffer 1 while playing half buffer 2
1352 <1>
1353 00013CE3 8B35[E8880100] <1> mov esi, [audio_p_buffer] ; phy addr of audio buff
1354 <1> ; 05/06/2024
1355 <1> ;shr ecx, 2 ; half buff size / 4
1356 <1> ;rep movsd
1357 00013CE9 D1E9 <1> shr ecx, 1 ; word aligned buffer
1358 00013CEB F366A5 <1> rep movsw
1359 <1>
1360 <1> ; switch flag value ;
1361 00013CEE 8035[FC880100]01 <1> xor byte [audio_flag], 1
1362 <1> ; 12/10/2017
1363 <1> ; [audio_flag] = 0 : Playing dma half buffer 2
1364 <1> ; ; Next buffer (to update) is dma half buff 1
1365 <1> ; = 1 : Playing dma half buffer 1
1366 <1> ; ; Next buffer (to update) is dma half buff 2
1367 <1> _ih4:
1368 <1> ; 28/05/2017
1369 <1> ;mov byte [audio_busy], 0 ; 09/10/2017
1370 <1> ;
1371 <1> ;pop edi
1372 <1> ;pop esi
1373 <1> ;pop ebx ; * must be restored !
1374 <1> ;pop ecx
1375 <1> ;pop edx
1376 <1> ;pop eax ; * must be restored !
1377 <1>
1378 00013CF5 C3 <1> retn
1379 <1>
1380 <1> channel_reset:
1381 <1> ; 06/08/2022 - TRDOS 386 kernel v2.0.5
1382 <1> ; 24/06/2017
1383 <1> ; 29/05/2017
1384 <1> ; 23/03/2017
1385 <1> ; 14/11/2016 - Erdogan Tan
1386 <1> ; 12/11/2016 - Erdogan Tan (Ref: KolibriOS, vt823x.asm)
1387 <1> ;mov edx, VIA_REG_OFFSET_CONTROL
1388 <1> ; 06/08/2022
1389 00013CF6 29D2 <1> sub edx, edx
1390 00013CF8 8201 <1> mov dl, VIA_REG_OFFSET_CONTROL
1391 <1> ;mov eax, VIA_REG_CTRL_PAUSE + VIA_REG_CTRL_TERMINATE + VIA_REG_CTRL_RESET
1392 00013CFA B848000000 <1> mov eax, VIA_REG_CTRL_PAUSE + VIA_REG_CTRL_TERMINATE ; 24/06/2017
1393 00013CFF E8A2FEFFFF <1> call ctrl_io_w8
1394 <1>
1395 <1> ;mov edx, VIA_REG_OFFSET_CONTROL
1396 <1> ;call ctrl_io_r8
1397 <1>
1398 <1> ; wait for 50 ms
1399 <1> ;mov ecx, 160 ; (200*0.25 ms) ; 29/05/2017
1400 <1> ; 06/08/2022
1401 00013D04 31C9 <1> xor ecx, ecx
1402 00013D06 81A0 <1> mov cl, 160
1403 <1> _ch_rst_wait:
1404 00013D08 E861FEFFFF <1> call delay1_4ms
1405 00013D0D 49 <1> dec ecx
1406 00013D0E 75F8 <1> jnz short _ch_rst_wait
1407 <1>
1408 <1> ; disable interrupts
1409 <1> ;mov edx, VIA_REG_OFFSET_CONTROL
1410 <1> ; 06/08/2022
1411 00013D10 29D2 <1> sub edx, edx
1412 00013D12 B201 <1> mov dl, VIA_REG_OFFSET_CONTROL
1413 00013D14 31C0 <1> xor eax, eax
1414 00013D16 E88BFEFFFF <1> call ctrl_io_w8
1415 <1>
1416 <1> ; clear interrupts
1417 <1> ;mov edx, VIA_REG_OFFSET_STATUS
1418 <1> ; 06/08/2022
1419 00013D1B 29D2 <1> sub edx, edx
1420 <1> ;mov dl, VIA_REG_OFFSET_STATUS ; 0
1421 <1> ; edx = 0
1422 <1> ;mov eax, 3
1423 <1> ; 06/08/2022
1424 00013D1D 29C0 <1> sub eax, eax
1425 00013D1F B003 <1> mov al, 3
1426 <1> ; eax = 3
1427 00013D21 E880FEFFFF <1> call ctrl_io_w8
1428 <1>
1429 <1> ;mov edx, VIA_REG_OFFSET_CURR_PTR
1430 <1> ;xor eax, eax
1431 <1> ;call ctrl_io_w32
1432 <1>
1433 00013D26 C3 <1> retn
1434 <1>
1435 <1> vt8233_stop: ; 22/04/2017
1436 00013D27 C605[0A890100]00 <1> mov byte [audio_play_cmd], 0 ; stop !

```

```

1437 <1> _t1p2:
1438 <1> ; 24/06/2017
1439 <1> ; finished with song, stop everything
1440 <1> ;mov al, VIA_REG_CTRL_INT
1441 <1> ;or al, VIA_REG_CTRL_TERMINATE
1442 <1> ;mov dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
1443 <1> ;call ctrl_io_w8
1444 <1>
1445 <1> ;call channel_reset
1446 <1> ;retn
1447 <1>
1448 00013D2E EBC6 <1> jmp short channel_reset
1449 <1>
1450 <1> set_vt8233_bdl: ; Set VT8237R Buffer Descriptor List
1451 <1> ; 04/06/2024 - TRDOS 386 v2.0.8
1452 <1> ; 06/08/2022 - TRDOS 386 v2.0.5
1453 <1> ; 22/07/2020 - TRDOS 386 v2.0.2
1454 <1> ; 28/05/2017
1455 <1> ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
1456 <1> ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
1457 <1>
1458 <1> ; eax = dma buffer address = [audio_DMA_buff]
1459 <1> ; ecx = dma buffer buffer size = [audio_dmabuff_size]
1460 <1>
1461 <1> ; 04/06/2024
1462 <1> ; ecx = DMA half buffer size (same as audio buffer size)
1463 <1>
1464 <1> ;shr ecx, 1 ; dma half buffer size
1465 00013D30 89CE <1> mov esi, ecx
1466 <1>
1467 00013D32 BF[10890100] <1> mov edi, audio_bdl_buff ; get BDL address
1468 <1> ;mov ecx, 32 / 2 ; make 32 entries in BDL
1469 <1> ; 06/08/2022
1470 00013D37 29C9 <1> sub ecx, ecx
1471 00013D39 B110 <1> mov cl, 16
1472 <1>
1473 00013D3B EB05 <1> jmp short s_vt8233_bdl1
1474 <1>
1475 <1> s_vt8233_bdl0:
1476 <1> ; set buffer descriptor 0 to start of data file in memory
1477 <1>
1478 00013D3D A1[F0880100] <1> mov eax, [audio_dma_buff] ; Physical address of DMA buffer
1479 <1>
1480 <1> s_vt8233_bdl1:
1481 00013D42 AB <1> stosd ; store dmabuffer1 address
1482 <1>
1483 00013D43 89C2 <1> mov edx, eax
1484 <1>
1485 <1> ; VIA VT8235.PDF: (Page 110) (Erdogan Tan, 29/11/2016)
1486 <1> ;
1487 <1> ; Audio SGD Table Format
1488 <1> ; -----
1489 <1> ; 63 62 61-56 55-32 31-0
1490 <1> ; -- --
1491 <1> ; EOL FLAG -reserved- Base Base
1492 <1> ; ; Count Address
1493 <1> ; ; [23:0] [31:0]
1494 <1> ;
1495 <1> ; EOL: End Of Link.
1496 <1> ; 1 indicates this block is the last of the link.
1497 <1> ; If the channel "Interrupt on EOL" bit is set, then
1498 <1> ; an interrupt is generated at the end of the transfer.
1499 <1> ;
1500 <1> ; FLAG: Block Flag. If set, transfer pauses at the end of this
1501 <1> ; block. If the channel "Interrupt on FLAG" bit is set,
1502 <1> ; then an interrupt is generated at the end of this block.
1503 00013D45 89F0 <1> mov eax, esi ; DMA half buffer size
1504 <1>
1505 <1> ; 04/06/2024 - Erdogan Tan
1506 <1> ; NOTE: I have changed DMA half buffer size to
1507 <1> ; (word aligned) audio buffer size for smooth audio playing
1508 <1> ; (it was page border aligned before June 4, 2024)
1509 <1> ;
1510 <1> ; For example:
1511 <1> ; For 65416 bytes audio buffer (22kHz, 16bit samples)
1512 <1> ; (user's audio buff virtual addr is mapped to physical)
1513 <1> ; ((kernel's and user's audio buff pages are same))
1514 <1> ; 1) Memory allocation (of it) is 65536 bytes
1515 <1> ; 2) DMA half buffer size is 65416 bytes
1516 <1> ; (it would be 65536 bytes before this modification)
1517 <1> ; ((additional 140 bytes would cause to a noise))
1518 <1> ; 3) Total DMA buffer size is 131072 bytes
1519 <1> ; (the last 240 bytes will not be used for playing)
1520 <1>
1521 00013D47 01C2 <1> add edx, eax
1522 00013D49 0D00000040 <1> or eax, FLAG
1523 <1> ;or eax, EOL
1524 00013D4E AB <1> stosd
1525 <1>
1526 <1> ; 2nd buffer:
1527 <1>
1528 00013D4F 89D0 <1> mov eax, edx ; Physical address of the 2nd half of DMA buffer
1529 00013D51 AB <1> stosd ; store dmabuffer2 address
1530 <1>
1531 <1> ; set length to [audio_dmabuff_size]/2
1532 <1> ; Set control (bits 31:16) to BUP, bits 15:0=number of samples
1533 <1> ;
1534 00013D52 89F0 <1> mov eax, esi ; DMA half buffer size
1535 <1> ; 22/07/2020
1536 <1> ;or eax, EOL
1537 00013D54 0D00000040 <1> or eax, FLAG
1538 00013D59 AB <1> stosd
1539 <1>
1540 00013D5A E2E1 <1> loop s_vt8233_bdl0
1541 <1>
1542 <1> ; 22/07/2020
1543 00013D5C 814FFC00000080 <1> or dword [edi-4], EOL
1544 <1>
1545 00013D63 C3 <1> retn
1546 <1>
1547 <1> vt8233_start_play:
1548 <1> ; 06/08/2022 - TRDOS 386 kernel v2.0.5
1549 <1> ; 01/09/2020
1550 <1> ; 22/07/2020
1551 <1> ; start to play audio data via VT8233 audio controller
1552 <1> ; 13/06/2017
1553 <1> ; 10/06/2017
1554 <1> ; 24/04/2017
1555 <1> ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
1556 <1> ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
1557 <1> ; write buffer descriptor list address
1558 <1>
1559 <1> ; Extended Audio Status (2Ah)
1560 00013D64 B82A000000 <1> mov eax, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah

```



```

1561 00013D69 E85CFEFFFF <1> call codec_read
1562 00013D6E 25FDFF0000 <1> and eax, 0FFFFh - 2 ; clear DRA (BIT1)
1563 <1> ;or eax, 1 ; set VRA (BIT0)
1564 <1> ;or eax, 5 ; VRA (BIT0) & S/PDIF (BIT2) ; 14/11/2016
1565 00013D73 0C05 <1> or al, 5
1566 <1> ; 01/09/2020
1567 <1> ;or eax, 3805h ; AD1980 (PRK, PRJ, PRI = 1 .. only front DAC)
1568 <1> ; 01/09/2020
1569 <1> ;mov edx, CODEC_EXT_AUDIO_CTRL_REG
1570 <1> ;cmp word [audio_freq], 0BB80h ; 48 kHz
1571 <1> ;jne short set_extd_audio_status_1
1572 <1> ;and al, 0FEh ; disable VRA bit (set sample rate to 48000 Hz)
1573 <1> ;jmp short set_extd_audio_status_2
1574 <1> ;set_extd_audio_status_1:
1575 00013D75 BA2A000000 <1> mov edx, CODEC_EXT_AUDIO_CTRL_REG
1576 00013D7A E86EFEFFFF <1> call codec_write
1577 <1> ;jc short cconfig_error
1578 <1>
1579 <1> set_sample_rate:
1580 <1> ;movzx eax, word [audio_freq]
1581 00013D7F 66A1[06890100] <1> mov ax, [audio_freq]
1582 00013D85 BA2C000000 <1> mov edx, CODEC_PCM_FRONT_DACRATE_REG ; 2Ch ; PCM Front DAC Rate
1583 <1> ;set_extd_audio_status_2:
1584 00013D8A E85EFEFFFF <1> call codec_write
1585 <1>
1586 <1> ; 01/09/2020
1587 <1> ; set AD1980 MCB register (Index 76h) to 0C00h
1588 <1> ; (CLDIS, HPSEL)
1589 <1> ;mov ax, 0C00h
1590 <1> ;mov edx, CODEC_MISC_CTRL_BITS_REG ; 76h
1591 <1> ; ; Miscellaneous Control Bit Register
1592 <1> ;call codec_write
1593 <1>
1594 <1>
1595 00013D8F B8[10890100] <1> mov eax, audio_bdl_buff
1596 <1>
1597 <1> ; 12/11/2016 - Erdogan Tan
1598 <1> ; (Ref: KolibriOS, vt823x.asm, 'create_primary_buff')
1599 <1> ;mov edx, VIADEV_PLAYBACK + VIA_REG_OFFSET_TABLE_PTR
1600 <1> ; 06/08/2022
1601 00013D94 29D2 <1> sub edx, edx
1602 <1> ;mov dl, VIADEV_PLAYBACK + VIA_REG_OFFSET_TABLE_PTR
1603 <1> ; edx = 0
1604 00013D96 E81DFEFFFF <1> call ctrl_io_w32
1605 <1>
1606 <1> ;call codec_check_ready
1607 <1>
1608 00013D9B 66BA0200 <1> mov dx, VIADEV_PLAYBACK + VIA_REG_OFS_PLAYBACK_VOLUME_L
1609 <1> ;mov eax, 2 ; 31
1610 00013D9F B01F <1> mov al, 31
1611 00013DA1 2A05[0C890100] <1> sub al, [audio_master_volume_l]
1612 00013DA7 E8FAFDFFFF <1> call ctrl_io_w8
1613 <1>
1614 <1> ;call codec_check_ready
1615 <1>
1616 00013DAC 66BA0300 <1> mov dx, VIADEV_PLAYBACK + VIA_REG_OFS_PLAYBACK_VOLUME_R
1617 <1> ;mov ax, 2 ; 31
1618 00013DB0 B01F <1> mov al, 31
1619 00013DB2 2A05[0D890100] <1> sub al, [audio_master_volume_r]
1620 00013DB8 E8E9FDFFFF <1> call ctrl_io_w8
1621 <1>
1622 <1> ;call codec_check_ready
1623 <1> ;
1624 <1> ;
1625 <1> ; All set. Let's play some music.
1626 <1> ;
1627 <1> ;
1628 <1> ;mov dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STOP_IDX
1629 <1> ;mov ax, VIA8233_REG_TYPE_16BIT or VIA8233_REG_TYPE_STEREO or 0xffff or 0xff000000
1630 <1> ;call ctrl_io_w32
1631 <1>
1632 <1> ;call codec_check_ready
1633 <1>
1634 <1> ; 08/12/2016
1635 <1> ; 07/10/2016
1636 <1> ;mov al, 1
1637 <1> ;mov al, 31
1638 <1> ; 22/07/2020
1639 00013DBD B0FF <1> mov al, 0FFh
1640 00013DBF E813000000 <1> call set_VT8233_LastValidIndex
1641 <1>
1642 00013DC4 C605[0A890100]01 <1> mov byte [audio_play_cmd], 1 ; play command (do not stop) !
1643 <1>
1644 <1> ; 22/07/2020
1645 <1> ;mov byte [audio_flag], 0 ; clear half buffer flag
1646 <1>
1647 <1> vt8233_play: ; continue to play
1648 <1> ; 22/04/2017
1649 <1> ;mov al, VIA_REG_CTRL_INT
1650 <1> ;or al, VIA_REG_CTRL_START
1651 <1> ;mov al, VIA_REG_CTRL_AUTOSTART + VIA_REG_CTRL_START
1652 <1> ; 22/07/2020
1653 00013DCB B0A1 <1> mov al, VIA_REG_CTRL_AUTOSTART + VIA_REG_CTRL_START + VIA_REG_CTRL_INT_FLAG
1654 <1>
1655 00013DCD 66BA0100 <1> mov dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
1656 00013DD1 E8D0FDFFFF <1> call ctrl_io_w8
1657 <1> ;call codec_check_ready
1658 <1> ;retn
1659 <1> ;jmp codec_check_ready
1660 00013DD6 C3 <1> retn
1661 <1>
1662 <1> ;input AL = index # to stop on
1663 <1> set_VT8233_LastValidIndex:
1664 <1> ; 06/08/2022 - TRDOS 386 kernel v2.0.5
1665 <1> ; 23/07/2020
1666 <1> ; 10/06/2017
1667 <1> ; 21/04/2017 (TRDOS 386 kernel, 'audio.s')
1668 <1> ; 24/03/2017 - 'PLAYER.COM' ('via_wav.asm' - 29/11/2016)
1669 <1> ; 19/11/2016
1670 <1> ; 14/11/2016 - Erdogan Tan (Ref: VIA VT8235.PDF, Page 110)
1671 <1> ; 12/11/2016 - Erdogan Tan
1672 <1> ; (Ref: KolibriOS, vt823x.asm, 'create_primary_buff')
1673 <1> ;push edx
1674 <1> ;push ax
1675 00013DD7 50 <1> push eax ; 23/07/2020
1676 <1> ;push ecx
1677 00013DD8 0FB705[06890100] <1> movzx eax, word [audio_freq] ; Hertz
1678 00013DDF BA00001000 <1> mov edx, 100000h ; 2^20 = 1048576
1679 00013DE4 F7E2 <1> mul edx
1680 00013DE6 B980BB0000 <1> mov ecx, 48000
1681 00013DEB F7F1 <1> div ecx
1682 <1> ;and eax, 0FFFFFFh
1683 <1> ;pop ecx
1684 <1> ;pop dx

```

```

1685 00013DED 5A          <1>    pop     edx ; 23/07/2020
1686 00013DEE C1E218      <1>    shl     edx, 24 ; STOP Index Setting: Bit 24 to 31
1687 00013DF1 09D0        <1>    or      eax, edx
1688                                <1>    ; 19/11/2016
1689 00013DF3 803D[04890100]10 <1>    cmp     byte [audio_bps], 16
1690 00013DFA 7505          <1>    jne     short sLVI_1
1691 00013DFC 0D00002000    <1>    or      eax, VIA8233_REG_TYPE_16BIT
1692                                <1>    sLVI_1:
1693 00013E01 803D[05890100]02 <1>    cmp     byte [audio_stmo], 2
1694 00013E08 7505          <1>    jne     short sLVI_2
1695 00013E0A 0D00001000    <1>    or      eax, VIA8233_REG_TYPE_STEREO
1696                                <1>    sLVI_2:
1697                                <1>    ;mov     edx, VIADEV_PLAYBACK + VIA_REG_OFFSET_STOP_IDX
1698                                <1>    ; 06/08/2022
1699 00013E0F 29D2          <1>    sub     edx, edx
1700 00013E11 B208          <1>    mov     dl, VIADEV_PLAYBACK + VIA_REG_OFFSET_STOP_IDX
1701 00013E13 E8A0FDFFFF    <1>    call    ctrl_io_w32
1702                                <1>    ;call    codec_check_ready
1703                                <1>    ;pop     edx
1704 00013E18 C3            <1>    retn
1705                                <1>
1706                                <1>    vt8233_pause: ; pause
1707                                <1>    ; 10/06/2017
1708                                <1>    ; 22/04/2017
1709                                <1>    ;mov     al, VIA_REG_CTRL_INT
1710                                <1>    ;or al, VIA_REG_CTRL_PAUSE
1711                                <1>    ; 23/07/2020
1712 00013E19 B029          <1>    mov     al, VIA_REG_CTRL_PAUSE+VIA_REG_CTRL_INT_FLAG+VIA_REG_CTRL_AUTOSTART
1713                                <1>
1714 00013E1B 66BA0100      <1>    mov     dx, VIADEV_PLAYBACK + VIA_REG_OFFSET_CONTROL
1715 00013E1F E882FDFFFF    <1>    call    ctrl_io_w8
1716                                <1>    ;call    codec_check_ready
1717                                <1>    ;retn
1718                                <1>    ;jmp     codec_check_ready
1719 00013E24 C3            <1>    retn
1720                                <1>
1721                                <1>    vt8233_reset:
1722                                <1>    ; 22/04/2017
1723                                <1>    ; reset VT8237R (vt8233) Audio Controller
1724                                <1>    ;cmp     byte [audio_play_cmd], 1
1725                                <1>    ;jna     short vt8233_rst_0
1726 00013E25 C605[0A890100]00 <1>    mov     byte [audio_play_cmd], 0 ; stop !
1727                                <1>    vt8233_rst_0:
1728                                <1>    call    reset_codec
1729 00013E31 720A          <1>    jc      short vt8233_rst_1 ; codec error !
1730                                <1>    ; eax = 1
1731 00013E33 E852FDFFFF    <1>    call    codec_io_w16 ; w32
1732 00013E38 E8B9FEFFFF    <1>    call    channel_reset
1733                                <1>    vt8233_rst_1:
1734                                <1>    vt8233_vol_1: ; 06/08/2022
1735 00013E3D C3            <1>    retn
1736                                <1>
1737                                <1>    vt8233_volume:
1738                                <1>    ; set VT8237R (vt8233) sound volume level
1739                                <1>    ; 06/08/2022 - TRDOS 386 v2.0.5
1740                                <1>    ; 24/04/2017
1741                                <1>    ; 22/04/2017
1742                                <1>    ; b1 = component (0 = master/playback/lineout volume)
1743                                <1>    ; c1 = left channel volume level (0 to 31)
1744                                <1>    ; ch = right channel volume level (0 to 31)
1745                                <1>
1746 00013E3E 08DB          <1>    or      b1, b1
1747 00013E40 75FB          <1>    jnz     short vt8233_vol_1 ; temporary !
1748 00013E42 66B81F1F      <1>    mov     ax, 1F1Fh ; 31,31
1749 00013E46 38C1          <1>    cmp     cl, al
1750 00013E48 77F3          <1>    ja      short vt8233_vol_1 ; temporary !
1751 00013E4A 38E5          <1>    cmp     ch, ah
1752 00013E4C 77EF          <1>    ja      short vt8233_vol_1 ; temporary !
1753 00013E4E 66890D[0C890100] <1>    mov     [audio_master_volume], cx
1754 00013E55 6629C8        <1>    sub     ax, cx
1755                                <1>    ;mov     edx, CODEC_MASTER_VOL_REG ; 02h ; Line Out
1756                                <1>    ; 06/08/2022
1757 00013E58 29D2          <1>    sub     edx, edx
1758 00013E5A B202          <1>    mov     dl, CODEC_MASTER_VOL_REG ; 02h ; Line Out
1759                                <1>    ; 06/08/2022
1760 00013E5C E98CFDFFFF    <1>    jmp     codec_write
1761                                <1>    ;call    codec_write
1762                                <1>    ;vt8233_vol_1:
1763                                <1>    ;retn
1764                                <1>
1765                                <1>    ; CODE for SOUND BLASTER 16
1766                                <1>
1767                                <1>    DetectSB:
1768                                <1>    ; 06/08/2022 - TRDOS 386 v2.0.5
1769                                <1>    ; 24/04/2017
1770                                <1>    ;pushad
1771                                <1>    ScanPort:
1772                                <1>    ; 06/08/2022
1773 00013E61 66BB1002      <1>    mov     bx, 0210h ; start scanning ports
1774                                <1>    ; 210h, 220h, .. 260h
1775                                <1>    ; 06/08/2022
1776 00013E65 31C9          <1>    xor     ecx, ecx
1777 00013E67 88FE          <1>    mov     dh, bh
1778                                <1>    ResetDSP:
1779                                <1>    ;mov     dx, bx ; try to reset the DSP.
1780                                <1>    ;add     dx, 06h
1781                                <1>    ; 06/08/2022
1782 00013E69 88DA          <1>    mov     dl, b1
1783 00013E6B 80C206        <1>    add     dl, 06h
1784                                <1>
1785 00013E6E B001          <1>    mov     al, 1
1786 00013E70 EE            <1>    out     dx, al
1787                                <1>
1788 00013E71 EC            <1>    in      al, dx
1789 00013E72 EC            <1>    in      al, dx
1790 00013E73 EC            <1>    in      al, dx
1791 00013E74 EC            <1>    in      al, dx
1792                                <1>
1793 00013E75 30C0          <1>    xor     al, al
1794 00013E77 EE            <1>    out     dx, al
1795                                <1>
1796                                <1>    ;add     dx, 08h
1797                                <1>    ; 06/08/2022
1798 00013E78 80C208        <1>    add     dl, 08h
1799                                <1>    ;mov     cx, 100
1800 00013E7B B164          <1>    mov     cl, 100
1801                                <1>    waitID:
1802 00013E7D EC            <1>    in      al, dx
1803 00013E7E 08C0          <1>    or      al, al
1804 00013E80 7804          <1>    js      short GetID
1805 00013E82 E2F9          <1>    loop    waitID
1806 00013E84 EB0D          <1>    jmp     short NextPort
1807                                <1>    GetID:
1808                                <1>    ;sub     dx, 04h

```

```

1809      <1>      ; 06/08/2022
1810 00013E86 80EA04 <1>      sub    dl, 04h
1811 00013E89 EC     <1>      in     al, dx
1812 00013E8A 3CAA   <1>      cmp    al, 0AAh
1813 00013E8C 740F   <1>      je     short Found
1814      <1>      ;add    dx, 04h
1815      <1>      ; 06/08/2022
1816 00013E8E 80C204 <1>      add    dl, 04h
1817 00013E91 E2EA   <1>      loop   waitID
1818      <1>      NextPort:
1819      <1>      ;add    bx, 10h          ; if not response,
1820      <1>      ; 06/08/2022
1821 00013E93 80C310 <1>      add    b1, 10h
1822      <1>      ;cmp    bx, 260h          ; try the next port.
1823 00013E96 80FB60 <1>      cmp    b1, 60h
1824 00013E99 76CE   <1>      jbe    short ResetDSP
1825 00013E9B F9     <1>      stc
1826 00013E9C C3     <1>      retn
1827      <1>      Found:
1828 00013E9D 66891D[D6880100] <1>      mov     [audio_io_base], bx ; SB Port Address Found!
1829      <1>      ScanIRQ:
1830      <1>      SetIrqs:
1831 00013EA4 28C0   <1>      sub    al, al ; 0
1832 00013EA6 A2[CC880100] <1>      mov     [IRQnum], al ; reset
1833 00013EAB A2[D2880100] <1>      mov     [audio_intr], al ; reset
1834      <1>
1835      <1>      ; ah > 0 -> set IRQ vector
1836      <1>      ; al = IRQ number
1837      <1>      ;mov    ax, 103h ; IRQ 3
1838      <1>      ;call   set_hardware_int_vector
1839      <1>      ;mov    ax, 104h ; IRQ 4
1840      <1>      ;call   set_hardware_int_vector
1841 00013EB0 66B80501 <1>      mov     ax, 105h ; IRQ 5
1842 00013EB4 E853DFFFFF <1>      call    set_hardware_int_vector
1843 00013EB9 66B80701 <1>      mov     ax, 107h ; IRQ 7
1844 00013EBD E84ADFFFFF <1>      call    set_hardware_int_vector
1845      <1>
1846 00013EC2 66B815[D6880100] <1>      mov     dx, [audio_io_base] ; tells to the SB to
1847      <1>      ;add    dx, 0Ch          ; generate a IRQ!
1848      <1>      ; 06/08/2022
1849 00013EC9 80C20C <1>      add    dl, 0Ch
1850      <1>      waitSb:
1851 00013ECC EC     <1>      in     al, dx
1852 00013ECD 08C0   <1>      or     al, al
1853 00013ECF 78FB   <1>      js     short waitSb
1854 00013ED1 80F2   <1>      mov     al, 0F2h
1855 00013ED3 EE     <1>      out    dx, al
1856      <1>
1857 00013ED4 31C9   <1>      xor     ecx, ecx          ; wait until IRQ level
1858      <1>      waitIRQ:
1859 00013ED6 A0[CC880100] <1>      mov     al, [IRQnum]
1860 00013EDB 3C00   <1>      cmp     al, 0 ; is changed or timeout.
1861 00013EDD 7706   <1>      ja     short IrqOk
1862 00013EDF 6649   <1>      dec     cx
1863 00013EE1 75F3   <1>      jnz     short waitIRQ
1864 00013EE3 E814   <1>      jmp     short RestoreIrqs
1865      <1>      IrqOk:
1866 00013EE5 A2[D2880100] <1>      mov     [audio_intr], al ; set
1867 00013EEA 66B815[D6880100] <1>      mov     dx, [audio_io_base]
1868      <1>      ;add    dx, 0Eh
1869      <1>      ; 06/08/2022
1870 00013EF1 80C20E <1>      add    dl, 0Eh
1871 00013EF4 EC     <1>      in     al, dx ; SB acknowledge.
1872 00013EF5 B020   <1>      mov     al, 20h
1873 00013EF7 E620   <1>      out    20h, al ; Hardware acknowledge.
1874      <1>
1875      <1>      RestoreIrqs:
1876      <1>      ; ah = 0 -> reset IRQ vector
1877      <1>      ; al = IRQ number
1878      <1>      ;mov    ax, 3 ; IRQ 3
1879      <1>      ;call   set_hardware_int_vector
1880      <1>      ;mov    ax, 4 ; IRQ 4
1881      <1>      ;call   set_hardware_int_vector
1882 00013EF9 66B80500 <1>      mov     ax, 5 ; IRQ 5
1883 00013EFD E80ADFFFFF <1>      call    set_hardware_int_vector
1884 00013F02 66B80700 <1>      mov     ax, 7 ; IRQ 7
1885 00013F06 E801DFFFFF <1>      call    set_hardware_int_vector
1886      <1>
1887 00013F0B 31D2   <1>      xor     edx, edx
1888 00013F0D 8915[D8880100] <1>      mov     [audio_dev_id], edx ; 0
1889 00013F13 8915[DC880100] <1>      mov     [audio_vendor], edx ; 0
1890      <1>      ; 22/11/2023
1891      <1>      ;mov    [audio_stats_cmd], edx ; 0
1892      <1>
1893      <1>      ;popad
1894      <1>
1895 00013F19 803D[D2880100]01 <1>      cmp     byte [audio_intr], 1 ; IRQ level was changed?
1896      <1>
1897 00013F20 C3     <1>      retn
1898      <1>
1899      <1>      %macro      SbOut 1
1900      <1>      %%wait:
1901      <1>      in     al, dx
1902      <1>      or     al, al
1903      <1>      js     short %%wait
1904      <1>      mov     al, %1
1905      <1>      out    dx, al
1906      <1>      %endmacro
1907      <1>
1908      <1>      SbInit_play:
1909      <1>      ; 28/01/2025 - TRDOS 386 kernel v2.0.10
1910      <1>      ; 06/08/2022 - TRDOS 386 kernel v2.0.5
1911      <1>      ; 22/10/2017
1912      <1>      ; 20/10/2017
1913      <1>      ; 06/10/2017
1914      <1>      ; 13/07/2017 - 09/08/2017
1915      <1>      ; 24/04/2017 - 15/05/2017 - 24/06/2017
1916      <1>      ;pushad
1917      <1>      SetBuffer:
1918      <1>      ;mov     byte [DmaFlag], 0
1919      <1>
1920 00013F21 8B1D[F0880100] <1>      mov     ebx, [audio_dma_buff] ; physical addr of DMA buff
1921 00013F27 89DF   <1>      mov     edi, ebx
1922      <1>      ;mov     ecx, [audio_dmabuff_size]
1923      <1>      ; 28/01/2025 (BugFix)
1924 00013F29 8B0D[F8880100] <1>      mov     ecx, [dma_hbuff_size]
1925      <1>      ;shl     ecx, 1 ; * 2 ; *!*
1926      <1>
1927 00013F2F 803D[04890100]10 <1>      cmp     byte [audio_bps], 16
1928 00013F36 752F   <1>      jne     short sbInit_0 ; set 8 bit DMA buffer
1929      <1>
1930      <1>      ; 09/08/2017
1931      <1>      ; convert byte count to word count
1932      <1>      ; 28/01/2025

```

```

1933      <1>      ;shr    ecx, 1 ; !***
1934      <1>
1935      00013F38 49      <1>      dec    ecx ; word count - 1
1936      <1>      ; convert byte offset to word offset
1937      00013F39 D1EB   <1>      shr    ebx, 1
1938      <1>
1939      <1>      ; 16 bit DMA buffer setting (DMA channel 5)
1940      00013F3B B005   <1>      mov    al, 05h ; set mask bit for channel 5 (4+1)
1941      00013F3D E6D4   <1>      out    0D4h, al
1942      <1>
1943      00013F3F 30C0   <1>      xor    al, al ; stops all DMA processes on selected channel
1944      00013F41 E6D8   <1>      out    0D8h, al ; clear selected channel register
1945      <1>
1946      00013F43 88D8   <1>      mov    al, bl ; byte 0 of DMA buffer offset in words (physical)
1947      00013F45 E6C4   <1>      out    0C4h, al ; DMA channel 5 port number
1948      <1>
1949      00013F47 88F8   <1>      mov    al, bh ; byte 1 of DMA buffer offset in words (physical)
1950      00013F49 E6C4   <1>      out    0C4h, al
1951      <1>
1952      <1>      ; 09/08/2017
1953      00013F4B C1EB0F <1>      shr    ebx, 15 ; complete 16 bit shift
1954      00013F4E 80E3FE <1>      and    bl, 0FEh ; clear bit 0 (not necessary, it will be ignored)
1955      <1>
1956      00013F51 88D8   <1>      mov    al, bl ; byte 2 of DMA buffer address (physical)
1957      00013F53 E68B   <1>      out    88h, al ; page register port addr for channel 5 ; 13/07/2017
1958      <1>
1959      00013F55 88C8   <1>      mov    al, cl ; low byte of DMA count - 1
1960      00013F57 E6C6   <1>      out    0C6h, al ; count register port addr for channel 1
1961      <1>
1962      00013F59 88E8   <1>      mov    al, ch ; high byte of DMA count - 1
1963      00013F5B E6C6   <1>      out    0C6h, al
1964      <1>
1965      <1>      ; channel 5, read, autoinitialized, single mode
1966      <1>      ;mov    al, 49h
1967      00013F5D B059   <1>      mov    al, 59h ; 06/10/2017
1968      00013F5F E6D6   <1>      out    0D6h, al ; DMA mode register port address
1969      <1>
1970      00013F61 B001   <1>      mov    al, 01h ; clear mask bit for channel 1
1971      00013F63 E6D4   <1>      out    0D4h, al ; DMA mask register port address
1972      <1>
1973      00013F65 EB2A   <1>      jmp     short ClearBuffer
1974      <1>
1975      <1>      sbInit_0:
1976      <1>      ; 28/01/2025
1977      00013F67 D1E1   <1>      shl    ecx, 1 ; half buffer size * 2 ; !***
1978      <1>
1979      00013F69 49      <1>      dec    ecx ; 09/08/2017
1980      <1>
1981      <1>      ; 8 bit DMA buffer setting (DMA channel 1)
1982      00013F6A B005   <1>      mov    al, 05h ; set mask bit for channel 1 (4+1)
1983      00013F6C E60A   <1>      out    0Ah, al ; DMA mask register
1984      <1>
1985      00013F6E 30C0   <1>      xor    al, al ; stops all DMA processes on selected channel
1986      00013F70 E60C   <1>      out    0Ch, al ; clear selected channel register
1987      <1>
1988      00013F72 88D8   <1>      mov    al, bl ; byte 0 of DMA buffer address (physical)
1989      00013F74 E602   <1>      out    02h, al ; DMA channel 1 port number
1990      <1>
1991      00013F76 88F8   <1>      mov    al, bh ; byte 1 of DMA buffer address (physical)
1992      00013F78 E602   <1>      out    02h, al
1993      <1>
1994      00013F7A C1EB10 <1>      shr    ebx, 16
1995      <1>
1996      00013F7D 88D8   <1>      mov    al, bl ; byte 2 of DMA buffer address (physical)
1997      00013F7F E683   <1>      out    83h, al ; page register port addr for channel 1
1998      <1>
1999      00013F81 88C8   <1>      mov    al, cl ; low byte of DMA count - 1
2000      00013F83 E603   <1>      out    03h, al ; count register port addr for channel 1
2001      <1>
2002      00013F85 88E8   <1>      mov    al, ch ; high byte of DMA count - 1
2003      00013F87 E603   <1>      out    03h, al
2004      <1>
2005      <1>      ; channel 1, read, autoinitialized, single mode
2006      <1>      ;mov    al, 49h
2007      00013F89 B059   <1>      mov    al, 59h ; 06/10/2017
2008      00013F8B E60B   <1>      out    0Bh, al ; DMA mode register port address
2009      <1>
2010      00013F8D B001   <1>      mov    al, 01h ; clear mask bit for channel 1
2011      00013F8F E60A   <1>      out    0Ah, al ; DMA mask register port address
2012      <1>
2013      <1>      ClearBuffer:
2014      <1>      ;mov    edi, [audio_dma_buff]
2015      <1>      ;mov    ecx, [audio_dmabuff_size]
2016      <1>      ;inc    ecx
2017      <1>      ;mov    al, 80h
2018      <1>      ;cld
2019      <1>      ;rep    stosb
2020      <1>      SetIrq:
2021      <1>      ;mov    ebx, SBIrqhandler
2022      <1>      ;mov    al, [audio_intr] ; IRQ number
2023      <1>      ;call   set_dev_IRQ_service
2024      <1>      ; ; SETUP (audio) INTERRUPT CALLBACK SERVICE
2025      <1>      ;mov    bl, [audio_intr] ; IRQ number
2026      <1>      ;mov    bh, [audio_cb_mode]
2027      <1>      ;inc    bh ; 1 = Signal Response Byte method (fixed value)
2028      <1>      ; ; 2 = Callback service method
2029      <1>      ; ; 3 = Auto Increment S.R.B. method
2030      <1>      ;mov    cl, [audio_srb]
2031      <1>      ;mov    edx, [audio_cb_addr]
2032      <1>      ;mov    al, [audio_user]
2033      <1>      ;call   set_irq_callback_service
2034      <1>      ResetDsp:
2035      00013F91 668B15[D6880100] <1>      mov    dx, [audio_io_base]
2036      <1>      ;add    dx, 06h
2037      <1>      ; 06/08/2022
2038      00013F98 80C206 <1>      add    dl, 06h
2039      00013F9B B001   <1>      mov    al, 1
2040      00013F9D EE      <1>      out    dx, al
2041      <1>
2042      00013F9E EC      <1>      in     al, dx
2043      00013F9F EC      <1>      in     al, dx
2044      00013FA0 EC      <1>      in     al, dx
2045      00013FA1 EC      <1>      in     al, dx
2046      <1>
2047      00013FA2 30C0   <1>      xor    al, al
2048      00013FA4 EE      <1>      out    dx, al
2049      <1>
2050      <1>      ;mov    cx, 100
2051      <1>      ; 06/08/2022
2052      00013FA5 29C9   <1>      sub    ecx, ecx
2053      00013FA7 B164   <1>      mov    cl, 100
2054      00013FA9 28E4   <1>      sub    ah, ah ; 0
2055      <1>      waitId:
2056      00013FAB 668B15[D6880100] <1>      mov    dx, [audio_io_base]

```

```

2057      <1>      ;add    dx, 0Eh
2058      <1>      ; 06/08/2022
2059 00013FB2 80C20E      <1>      add     dl, 0Eh
2060 00013FB5 EC          <1>      in      al, dx
2061 00013FB6 08C0        <1>      or      al, al
2062 00013FB8 7807        <1>      js      short sb_GetId
2063 00013FBA E2EF        <1>      loop    waitId
2064 00013FBC E9B1000000   <1>      jmp     sb_Exit
2065      <1>      sb_GetId:
2066 00013FC1 668B15[D6880100] <1>      mov     dx, [audio_io_base]
2067      <1>      ;add     dx, 0Ah
2068      <1>      ; 06/08/2022
2069 00013FC8 80C20A      <1>      add     dl, 0Ah
2070 00013FCB EC          <1>      in      al, dx
2071 00013FCC 3CAA        <1>      cmp     al, 0AAh
2072 00013FCE 7407        <1>      je      short Sbok
2073 00013FD0 E2D9        <1>      loop    waitId
2074 00013FD2 E99B000000   <1>      jmp     sb_Exit
2075      <1>      Sbok:
2076 00013FD7 668B15[D6880100] <1>      mov     dx, [audio_io_base]
2077      <1>      ;add     dx, 0Ch
2078      <1>      ; 06/08/2022
2079 00013FDE 80C20C      <1>      add     dl, 0Ch
2080      <1>      SbOut   0D1h ; Turn on speaker
1900      <2>      %%wait:
1901 00013FE1 EC          <2>      in      al, dx
1902 00013FE2 08C0        <2>      or      al, al
1903 00013FE4 78FB        <2>      js      short %%wait
1904 00013FE6 B0D1        <2>      mov     al, %1
1905 00013FE8 EE          <2>      out     dx, al
2081      <1>      SbOut   41h ; 8h bit or 16 bit transfer
1900      <2>      %%wait:
1901 00013FE9 EC          <2>      in      al, dx
1902 00013FEA 08C0        <2>      or      al, al
1903 00013FEC 78FB        <2>      js      short %%wait
1904 00013FEE B041        <2>      mov     al, %1
1905 00013FF0 EE          <2>      out     dx, al
2082 00013FF1 668B1D[06890100] <1>      mov     bx, [audio_freq] ; sampling rate (Hz)
2083      <1>      SbOut   bh ; sampling rate high byte
1900      <2>      %%wait:
1901 00013FF8 EC          <2>      in      al, dx
1902 00013FF9 08C0        <2>      or      al, al
1903 00013FFB 78FB        <2>      js      short %%wait
1904 00013FFD 88F8        <2>      mov     al, %1
1905 00013FFF EE          <2>      out     dx, al
2084      <1>      SbOut   b1 ; sampling rate low byte
1900      <2>      %%wait:
1901 00014000 EC          <2>      in      al, dx
1902 00014001 08C0        <2>      or      al, al
1903 00014003 78FB        <2>      js      short %%wait
1904 00014005 88D8        <2>      mov     al, %1
1905 00014007 EE          <2>      out     dx, al
2085      <1>
2086      <1>      ; 22/05/2017
2087 00014008 E8BD000000   <1>      call    sb16_volume_initial ; 15/05/2017
2088      <1>      ; 20/05/2017
2089      <1>      ;call   sb16_volume
2090      <1>
2091      <1>      StartDma:
2092      <1>      ; autoinitialized mode
2093 0001400D 803D[04890100]10 <1>      cmp     byte [audio_bps], 16 ; 16 bit samples
2094 00014014 7411        <1>      je      short sb_play_1
2095      <1>      ; 8 bit samples
2096 00014016 66BBC600      <1>      mov     bx, 0C6h ; 8 bit output (0C6h)
2097 0001401A 803D[05890100]02 <1>      cmp     byte [audio_stmo], 2 ; 1 = mono, 2 = stereo
2098 00014021 7214        <1>      jb      short sb_play_2
2099 00014023 B720        <1>      mov     bh, 20h ; 8 bit stereo (20h)
2100 00014025 EB10        <1>      jmp     short sb_play_2
2101      <1>      sb_play_1:
2102      <1>      ; 16 bit samples
2103 00014027 66BBB610      <1>      mov     bx, 10B6h ; 16 bit output (0B6h)
2104 0001402B 803D[05890100]02 <1>      cmp     byte [audio_stmo], 2 ; 1 = mono, 2 = stereo
2105 00014032 7203        <1>      jb      short sb_play_2
2106 00014034 80C720      <1>      add     bh, 20h ; 16 bit stereo (30h)
2107      <1>      sb_play_2:
2108      <1>      ; PCM output (8/16 bit mono autoinitialized transfer)
2109      <1>      SbOut   b1 ; bCommand
1900      <2>      %%wait:
1901 00014037 EC          <2>      in      al, dx
1902 00014038 08C0        <2>      or      al, al
1903 0001403A 78FB        <2>      js      short %%wait
1904 0001403C 88D8        <2>      mov     al, %1
1905 0001403E EE          <2>      out     dx, al
2110      <1>      SbOut   bh ; bMode
1900      <2>      %%wait:
1901 0001403F EC          <2>      in      al, dx
1902 00014040 08C0        <2>      or      al, al
1903 00014042 78FB        <2>      js      short %%wait
1904 00014044 88F8        <2>      mov     al, %1
1905 00014046 EE          <2>      out     dx, al
2111 00014047 8B1D[F4880100] <1>      mov     ebx, [audio_dmabuff_size] ; 15/05/2017
2112 0001404D D1EB        <1>      shr     ebx, 1 ; half buffer size
2113      <1>      ; 20/10/2017
2114 0001404F 803D[04890100]10 <1>      cmp     byte [audio_bps], 16 ; 16 bit DMA
2115 00014056 7502        <1>      jne     short sb_play_3
2116 00014058 D1EB        <1>      shr     ebx, 1 ; byte count to word count
2117      <1>      sb_play_3:
2118      <1>      ;dec     bx ; wBlkSize is one less than the actual size
2119      <1>      ; 06/08/2022
2120 0001405A 4B          <1>      dec     ebx
2121      <1>      SbOut   b1
1900      <2>      %%wait:
1901 0001405B EC          <2>      in      al, dx
1902 0001405C 08C0        <2>      or      al, al
1903 0001405E 78FB        <2>      js      short %%wait
1904 00014060 88D8        <2>      mov     al, %1
1905 00014062 EE          <2>      out     dx, al
2122      <1>      SbOut   bh
1900      <2>      %%wait:
1901 00014063 EC          <2>      in      al, dx
1902 00014064 08C0        <2>      or      al, al
1903 00014066 78FB        <2>      js      short %%wait
1904 00014068 88F8        <2>      mov     al, %1
1905 0001406A EE          <2>      out     dx, al
2123      <1>
2124 0001406B C605[0A890100]01 <1>      mov     byte [audio_play_cmd], 1 ; playing !
2125      <1>
2126      <1>      ;; Set voice and master volumes
2127      <1>      ;mov     dx, [audio_io_base]
2128      <1>      ;add     dl, 4 ; Mixer chip Register Address Port
2129      <1>      ;SbOut   30h ; select Master Volume Register (L)
2130      <1>      ;inc     dl ; Mixer chip Register Data Port
2131      <1>      ;SbOut   0F8h ; Max. volume value is 31 (31*8)
2132      <1>      ;dec     dl

```

```

2133      <1>      ;SbOut 31h ; select Master Volume Register (R)
2134      <1>      ;inc    dl
2135      <1>      ;SbOut 0F8h ; Max. volume value is 31 (31*8)
2136      <1>      ;dec    dl
2137      <1>      ;SbOut 32h ; select Voice Volume Register (L)
2138      <1>      ;inc    dl
2139      <1>      ;SbOut 0F8h ; Max. volume value is 31 (31*8)
2140      <1>      ;dec    dl
2141      <1>      ;SbOut 33h ; select Voice Volume Register (R)
2142      <1>      ;inc    dl
2143      <1>      ;SbOut 0F8h ; Max. volume value is 31 (31*8)
2144      <1>      ;
2145      <1>      ;dec    dl
2146      <1>      ;SbOut 44h ; select Treble Register (L)
2147      <1>      ;inc    dl
2148      <1>      ;SbOut 0F0h ; Max. Treble value is 15 (15*16)
2149      <1>      ;dec    dl
2150      <1>      ;SbOut 45h ; select Treble Register (R)
2151      <1>      ;inc    dl
2152      <1>      ;SbOut 0F0h ; Max. Treble value is 15 (15*16)
2153      <1>      ;dec    dl
2154      <1>      ;SbOut 46h ; select Bass Register (L)
2155      <1>      ;inc    dl
2156      <1>      ;SbOut 0F0h ; Max. Bass value is 15 (15*16)
2157      <1>      ;dec    dl
2158      <1>      ;SbOut 47h ; select Bass Register (R)
2159      <1>      ;inc    dl
2160      <1>      ;SbOut 0F0h ; Max. Bass value is 15 (15*16)
2161      <1>
2162      <1>      sb_Exit:
2163      <1>      ;popad
2164      <1>      ;ret
2165      <1>
2166      <1>      sb16_int_handler:
2167      <1>      ; Interrupt Handler for Sound Blaster 16 Audio Card
2168      <1>      ; Note: called by 'dev_IRQ_service'
2169      <1>      ; 28/01/2025
2170      <1>      ; TRDOS 386 Kernel v2.0.10
2171      <1>      ; 20/10/2017
2172      <1>      ; 12/10/2017
2173      <1>      ; 10/10/2017
2174      <1>      ; 12/05/2017, 09/10/2017
2175      <1>      ; 24/04/2017 (TRDOS 386 kernel, 'audio.s')
2176      <1>      ; 10/03/2017 - 'PLAYWAV.PRG' ('playwav.s')
2177      <1>
2178      <1>      ;push    eax ; * must be saved !
2179      <1>      ;push    ebx ; * must be saved !
2180      <1>      ;push    ecx
2181      <1>      ;push    edx
2182      <1>      ;push    esi
2183      <1>      ;push    edi
2184      <1>
2185      <1>      mov     dx, [audio_io_base]
2186      <1>      ; 20/10/2017
2187      <1>      add     dl, 0Fh ; 2xFh (DSP 16 bit intr ack)
2188      <1>      cmp     byte [audio_bps], 16
2189      <1>      je      short sb_irq_16bit_ack
2190      <1>      sb_irq_8bit_ack:
2191      <1>      dec     dl ; 2xEh (DSP 8 bit intr ack)
2192      <1>      sb_irq_16bit_ack:
2193      <1>      in      al, dx
2194      <1>
2195      <1>      ;cmp     byte [audio_busy], 0
2196      <1>      ;ja      short sb_irq_h3
2197      <1>
2198      <1>      ;mov     byte [audio_busy], 1
2199      <1>
2200      <1>      cmp     byte [audio_play_cmd], 1
2201      <1>      jnb     short sb_irq_h1
2202      <1>      sb_irq_h0:
2203      <1>      call    sb16_stop
2204      <1>      jmp     short sb_irq_h3
2205      <1>      sb_irq_h1:
2206      <1>      ;call    sb16_tuneloop
2207      <1>      ; 09/10/2017
2208      <1>      sb16_tuneloop:
2209      <1>      mov     edi, [audio_dma_buff]
2210      <1>      ;mov     ecx, [audio_dmabuff_size]
2211      <1>      ;shr     ecx, 1 ; dma buff size / 2 = half buffer size
2212      <1>      ; 28/01/2025 (BugFix)
2213      <1>      mov     ecx, [dma_hbuff_size]
2214      <1>
2215      <1>      ; 22/05/2017
2216      <1>      test    byte [audio_flag], 1 ; Current flag value
2217      <1>      jz      short sb_tlp1 ; EOL (Half Buffer 1 must be filled)
2218      <1>      ; FLAG (Half Buffer 2 must be filled)
2219      <1>      add     edi, ecx
2220      <1>      ; 15/05/2017
2221      <1>      sb_tlp1:
2222      <1>      mov     esi, [audio_p_buffer] ; phy addr of audio buff
2223      <1>      ;rep     movsb
2224      <1>      shr     ecx, 2 ; half buff size / 4
2225      <1>      rep     movsd
2226      <1>      ;ret
2227      <1>
2228      <1>      ; 10/10/2017
2229      <1>      ; switch flag value
2230      <1>      xor     byte [audio_flag], 1
2231      <1>
2232      <1>      ; 12/10/2017
2233      <1>      ; [audio_flag] = 0 : Playing dma half buffer 2 (odd intr count)
2234      <1>      ; Next buffer (to update) is dma half buff 1
2235      <1>      ; = 1 : Playing dma half buffer 1 (even intr count)
2236      <1>      ; Next buffer (to update) is dma half buff 2
2237      <1>
2238      <1>      sb_irq_h3:
2239      <1>      ;mov     byte [audio_busy], 0
2240      <1>
2241      <1>      ;pop     edi
2242      <1>      ;pop     esi
2243      <1>      ;pop     edx
2244      <1>      ;pop     ecx
2245      <1>      ;pop     ebx ; * must be restored !
2246      <1>      ;pop     eax ; * must be restored !
2247      <1>
2248      <1>      ;ret
2249      <1>
2250      <1>      sb16_volume:
2251      <1>      ; 06/08/2022 (TRDOS 386 v2.0.5)
2252      <1>      ; 22/10/2017
2253      <1>      ; mov     [audio_master_volume_l], cl
2254      <1>      ; mov     [audio_master_volume_h], ch
2255      <1>      mov     [audio_master_volume], cx
2256      <1>      sb16_volume_initial:

```

```

2257 <1> ;push dx ; DX (port address) must be saved
2258 <1> ; 06/08/2022
2259 000140CA 52 <1> push edx
2260 000140CB 668B15[D6880100] <1> mov dx, [audio_io_base]
2261 <1> ;add dx, 4 ; Mixer chip address port
2262 <1> ; 06/08/2022
2263 000140D2 80C204 <1> add dl, 4
2264 000140D5 B022 <1> mov al, 22h ; master volume
2265 000140D7 EE <1> out dx, al
2266 <1> ;inc dx
2267 <1> ; 06/08/2022
2268 000140D8 42 <1> inc edx
2269 000140D9 8A25[0C890100] <1> mov ah, [audio_master_volume_1]
2270 000140DF C0EC02 <1> shr ah, 2 ; 32 -> 8 level
2271 000140E2 C0E405 <1> shl ah, 5 ; bit 5 to 7
2272 000140E5 A0[0D890100] <1> mov al, [audio_master_volume_r]
2273 000140EA C0E802 <1> shr al, 2 ; 32 -> 8 level
2274 <1> ;and al, 0Fh
2275 000140ED D0E0 <1> shl al, 1 ; bit 1 to 3
2276 000140EF 08E0 <1> or al, ah
2277 000140F1 EE <1> out dx, al
2278 <1> ;pop dx ; DX (port address) must be restored
2279 <1> ; 06/08/2022
2280 000140F2 5A <1> pop edx
2281 000140F3 C3 <1> retn
2282 <1>
2283 <1> sb16_pause:
2284 <1> ; 06/08/2022 (TRDOS 386 v2.0.5)
2285 000140F4 668B15[D6880100] <1> mov dx, [audio_io_base]
2286 <1> ;add dx, 0Ch ; Command & Data Port
2287 <1> ; 06/08/2022
2288 000140FB 80C20C <1> add dl, 0Ch
2289 000140FE 803D[04890100]10 <1> cmp byte [audio_bps], 16 ; 16 bit samples
2290 00014105 7404 <1> je short sb_pause_1
2291 <1> ; 8 bit samples
2292 00014107 B3D0 <1> mov bl, 0D0h ; 8 bit DMA mode
2293 00014109 EB02 <1> jmp short sb_pause_2
2294 <1> sb_pause_1:
2295 <1> ; 16 bit samples
2296 0001410B B3D5 <1> mov bl, 0D5h ; 16 bit DMA mode
2297 <1> sb_pause_2:
2298 <1> SbOut bl ; bCommand
2299 <2> %%wait:
2300 00014115 C3 <2> in al, dx
2301 <2> or al, al
2302 <2> js short %%wait
2303 <2> mov al, %1
2304 <2> out dx, al
2305 <1> sb_pause_3:
2306 <1> retn
2307 <1>
2308 <1> sb16_continue:
2309 <1> ; 06/08/2022 (TRDOS 386 v2.0.5)
2310 00014116 668B15[D6880100] <1> mov dx, [audio_io_base]
2311 <1> ;add dx, 0Ch ; Command & Data Port
2312 <1> ; 06/08/2022
2313 0001411D 80C20C <1> add dl, 0Ch
2314 00014120 803D[04890100]10 <1> cmp byte [audio_bps], 16 ; 16 bit samples
2315 00014127 7404 <1> je short sb_cont_1
2316 <1> ; 8 bit samples
2317 00014129 B3D4 <1> mov bl, 0D4h ; 8 bit DMA mode
2318 0001412B EB02 <1> jmp short sb_cont_2
2319 <1> sb_cont_1:
2320 <1> ; 16 bit samples
2321 0001412D B3D6 <1> mov bl, 0D6h ; 16 bit DMA mode
2322 <1> sb_cont_2:
2323 <1> SbOut bl ; bCommand
2324 <2> %%wait:
2325 0001412F EC <2> in al, dx
2326 00014130 08C0 <2> or al, al
2327 00014132 78FB <2> js short %%wait
2328 00014134 88D8 <2> mov al, %1
2329 00014136 EE <2> out dx, al
2330 <1> sb_cont_3:
2331 00014137 C3 <1> retn
2332 <1>
2333 <1> sb16_stop:
2334 <1> ; 06/08/2022 (TRDOS 386 v2.0.5)
2335 <1> ; 24/04/2017
2336 00014138 803D[0A890100]00 <1> cmp byte [audio_play_cmd], 0
2337 0001413F 7647 <1> jna short sb16_stop_4
2338 <1>
2339 <1> ; 22/05/2017
2340 00014141 668B15[D6880100] <1> mov dx, [audio_io_base]
2341 <1> ;add dx, 0Ch
2342 <1> ; 06/08/2022
2343 00014148 80C20C <1> add dl, 0Ch
2344 <1>
2345 0001414B B3D9 <1> mov bl, 0D9h ; exit auto-initialize 16 bit transfer
2346 <1> ; stop autoinitialized DMA transfer mode
2347 0001414D 803D[04890100]10 <1> cmp byte [audio_bps], 16 ; 16 bit samples
2348 00014154 7402 <1> je short sb16_stop_1
2349 <1> ;mov bl, 0DAh ; exit auto-initialize 8 bit transfer
2350 00014156 FEC3 <1> inc bl
2351 <1> sb16_stop_1:
2352 <1> SbOut bl ; exit auto-initialize transfer command
2353 <2> %%wait:
2354 00014158 EC <2> in al, dx
2355 00014159 08C0 <2> or al, al
2356 0001415B 78FB <2> js short %%wait
2357 0001415D 88D8 <2> mov al, %1
2358 0001415F EE <2> out dx, al
2359 <1>
2360 00014160 30C0 <1> xor al, al ; stops all DMA processes on selected channel
2361 <1>
2362 00014162 803D[04890100]10 <1> cmp byte [audio_bps], 16 ; 16 bit samples
2363 00014169 7404 <1> je short sb16_stop_2
2364 0001416B E60C <1> out 0Ch, al ; clear selected channel register
2365 0001416D EB02 <1> jmp short sb16_stop_3
2366 <1>
2367 <1> sb16_stop_2:
2368 0001416F E6D8 <1> out 0D8h, al ; clear selected channel register
2369 <1>
2370 <1> sb16_stop_3:
2371 00014171 C605[0A890100]00 <1> mov byte [audio_play_cmd], 0 ; stop !
2372 <1> SbDone:
2373 <1> ;mov dx, [audio_io_base]
2374 <1> ;add dx, 0Ch
2375 <1> SbOut 0D0h
2376 <2> %%wait:
2377 00014178 EC <2> in al, dx
2378 00014179 08C0 <2> or al, al
2379 0001417B 78FB <2> js short %%wait
2380 0001417D B0D0 <2> mov al, %1

```

```

1905 0001417F EE      <2> out dx, al
2358                <1> sbOut 0D3h
1900                <2> %%wait:
1901 00014180 EC      <2> in al, dx
1902 00014181 08C0    <2> or al, al
1903 00014183 78FB    <2> js short %%wait
1904 00014185 B0D3    <2> mov al, %1
1905 00014187 EE      <2> out dx, al
2359                <1> sb16_stop_4:
2360 00014188 C3      <1> retn
2361                <1>
2362                <1> sb16_reset:
2363                <1> ; 06/08/2022 (TRDOS 386 v2.0.5)
2364                <1> ; 24/04/2017
2365 00014189 668B15[D6880100] <1> mov dx, [audio_io_base] ; try to reset the DSP.
2366                <1> ;add dx, 06h
2367                <1> ; 06/08/2022
2368 00014190 80C206    <1> add dl, 06h
2369 00014193 B001    <1> mov al, 1
2370 00014195 EE      <1> out dx, al
2371                <1>
2372 00014196 EC      <1> in al, dx
2373 00014197 EC      <1> in al, dx
2374 00014198 EC      <1> in al, dx
2375 00014199 EC      <1> in al, dx
2376                <1>
2377 0001419A 30C0    <1> xor al, al
2378 0001419C EE      <1> out dx, al
2379                <1>
2380                <1> ;add dx, 08h
2381                <1> ; 06/08/2022
2382 0001419D 80C208    <1> add dl, 08h
2383                <1> ;mov cx, 100
2384 000141A0 29C9    <1> sub ecx, ecx
2385 000141A2 B164    <1> mov cl, 100
2386                <1> sbrstwaitID:
2387 000141A4 EC      <1> in al, dx
2388 000141A5 08C0    <1> or al, al
2389 000141A7 7804    <1> js short sbrstGetID
2390 000141A9 E2F9    <1> loop sbrstwaitID
2391 000141AB F9      <1> stc
2392 000141AC C3      <1> retn
2393                <1> sbrstGetID:
2394                <1> ;sub dx, 04h
2395                <1> ; 06/08/2022
2396 000141AD 80EA04    <1> sub dl, 04h
2397 000141B0 EC      <1> in al, dx
2398 000141B1 3CAA    <1> cmp al, 0AAh
2399 000141B3 7405    <1> je short sb_rst_retn
2400                <1> ;add dx, 04h
2401                <1> ; 06/08/2022
2402 000141B5 80C204    <1> add dl, 04h
2403 000141B8 E2EA    <1> loop sbrstwaitID
2404                <1> sb_rst_retn:
2405 000141BA C3      <1> retn
2406                <1>
2407                <1> ac97_codec_config:
2408                <1> ; 06/06/2024
2409                <1> ; 04/06/2024
2410                <1> ; 03/06/2024
2411                <1> ; 02/06/2024
2412                <1> ; 01/06/2024 (TRDOS 386 v2.0.8)
2413                <1> ; 26/11/2023
2414                <1> ; 21/11/2023
2415                <1> ; 20/11/2023
2416                <1> ; 19/11/2023 (TRDOS 386 v2.0.7)
2417                <1> ; 10/06/2017
2418                <1> ; 05/06/2017
2419                <1> ; 29/05/2017
2420                <1> ; 28/05/2017 (TRDOS 386, 'audio.s')
2421                <1> ; 07/11/2016 (Erdogan Tan)
2422                <1> ; Derived from 'codecConfig' procedure in 'CODEC.ASM'
2423                <1> ; .wav player for DOS by Jeff Leyda (02/09/2002)
2424                <1>
2425                <1> ;; 'PLAYER.ASM'
2426                <1> ;; get ICH base address regs for mixer and bus master
2427                <1>
2428                <1> init_ac97_controller: ; 10/06/2017
2429 000141BB A1[D8880100] <1> mov eax, [audio_dev_id]
2430                <1> ;mov al, NAMBAR_REG
2431                <1> ;;call pciRegRead16 ; read PCI registers 10-11
2432                <1> ;call pciRegRead32
2433                <1> ;and dx, IO_ADDR_MASK ; mask off BIT0
2434                <1> ;;and edx, IO_ADDR_MASK
2435                <1>
2436                <1> ;mov [NAMBAR], dx ; save audio mixer base addr
2437                <1>
2438                <1> ;mov al, NABMBAR_REG
2439                <1> ;;call pciRegRead16
2440                <1> ;call pciRegRead32
2441                <1> ;and dx, 0FFC0h ; IO_ADDR_MASK
2442                <1> ;;and edx, 0FFC0h
2443                <1>
2444                <1> ;mov [NABMBAR], dx ; save bus master base addr
2445                <1>
2446                <1> ;mov eax, [audio_dev_id]
2447 000141C0 B004    <1> mov al, PCI_CMD_REG
2448                <1> ;call pciRegRead8 ; read PCI command register
2449 000141C2 E897F8FFFF <1> call pciRegRead16
2450 000141C7 80CA05    <1> or dl, IO_ENA+BM_ENA ; enable IO and bus master
2451                <1> ;call pciRegWrite8
2452 000141CA E8FAF8FFFF <1> call pciRegWrite16
2453                <1>
2454                <1> ; 'CODEC.ASM'
2455                <1>
2456                <1> ; enable codec, unmute stuff, set output rate
2457                <1> ; entry: [audio_freq] = desired sample rate
2458                <1>
2459                <1> ; mov dx, [NAMBAR]
2460                <1> ; add dx, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
2461                <1> ; in ax, dx
2462                <1> ; or ax, 1
2463                <1> ; out dx, ax ; Enable variable rate audio
2464                <1>
2465                <1> ; call delay1_4ms
2466                <1> ; call delay1_4ms
2467                <1> ; call delay1_4ms
2468                <1> ; call delay1_4ms
2469                <1>
2470                <1> ; mov ax, [audio_freq] ; sample rate
2471                <1>
2472                <1> ; mov dx, [NAMBAR]
2473                <1> ; add dx, CODEC_PCM_FRONT_DACRATE_REG ; 2Ch
2474                <1> ; out dx, ax ; out sample rate

```



```

2475 <1>
2476 <1> ; ;call delay1_4ms
2477 <1> ; ;call delay1_4ms
2478 <1> ; ;call delay1_4ms
2479 <1> ; ;call delay1_4ms
2480 <1>
2481 <1> ;mov dx, [NAMBAR] ; mixer base address
2482 <1> ;add dx, CODEC_RESET_REG ; reset register
2483 <1> ;mov ax, 42
2484 <1> ;out dx, ax ; reset
2485 <1>
2486 <1> ;mov dx, [NABMBAR] ; bus master base address
2487 <1> ;add dx, GLOB_STS_REG
2488 <1> ;mov ax, 2
2489 <1> ;out dx, ax
2490 <1>
2491 <1> ; 01/06/2024
2492 <1> ; 16/05/2024
2493 <1> ; 02/12/2023
2494 <1> ;call delay_100ms ; 29/05/2017
2495 <1>
2496 <1> ; 02/12/2023
2497 <1> ;call delay1_4ms
2498 <1> ;call delay1_4ms
2499 <1> ;call delay1_4ms
2500 <1> ;call delay1_4ms
2501 <1>
2502 <1> init_ac97_codec:
2503 <1> ; 26/11/2023
2504 <1> ; 19/11/2023
2505 <1> ; (playwav3.com, ac97_vra.asm, Erdogan Tan, 19/11/2023)
2506 <1> ; 10/06/2017
2507 <1> ; 29/05/2017
2508 <1> ; 28/05/2017 - Erdogan Tan (Ref: kolibriOS, intelac97.asm)
2509 <1>
2510 <1> ;; 23/11/2023 - temporary
2511 <1> ;push ebx
2512 <1> ;mov ebx, 0B8000h
2513 <1> ;mov al, '?'
2514 <1> ;mov ah, 4Eh
2515 <1> ;mov [ebx], ax
2516 <1> ;pop ebx
2517 <1>
2518 <1> ; 01/06/2024
2519 000141CF C605[148D0100]01 <1> mov byte [reset], 1
2520 <1>
2521 <1> ; 19/11/2023
2522 000141D6 BD28000000 <1> mov ebp, 40 ; 21/11/2023
2523 <1> _initc_1:
2524 <1> ; 26/05/2024
2525 000141DB 66BA3000 <1> mov dx, GLOB_STS_REG ; 30h
2526 000141DF 660315[D6880100] <1> add dx, [NABMBAR]
2527 000141E6 ED <1> in eax, dx
2528 <1>
2529 <1> ; 01/06/2024
2530 <1> ; 02/12/2023
2531 <1> ;call delay1_4ms
2532 <1>
2533 000141E7 83F8FF <1> cmp eax, 0FFFFFFFh ; -1
2534 000141EA 750A <1> jne short _initc_3
2535 <1> _initc_2:
2536 000141EC 4D <1> dec ebp ; 21/11/2023
2537 000141ED 7425 <1> jz short _ac97_codec_ready
2538 <1>
2539 000141EF E86DF9FFFF <1> call delay_100ms
2540 000141F4 EBE5 <1> jmp short _initc_1
2541 <1> _initc_3:
2542 000141F6 A900030010 <1> test eax, CTRL_ST_CREADY
2543 000141FB 7517 <1> jnz short _ac97_codec_ready
2544 <1>
2545 <1> ; 01/06/2024
2546 000141FD 803D[148D0100]00 <1> cmp byte [reset], 0
2547 00014204 76E6 <1> jna short _initc_2
2548 <1>
2549 00014206 E849030000 <1> call reset_ac97_codec
2550 <1> ;jc short _initc_2
2551 <1> ; 01/06/2024
2552 0001420B C605[148D0100]00 <1> mov byte [reset], 0
2553 <1> ; 26/11/2023
2554 00014212 EBD8 <1> jmp short _initc_2
2555 <1>
2556 <1> _ac97_codec_ready:
2557 00014214 668B15[D4880100] <1> mov dx, [NAMBAR]
2558 <1> ;add dx, 0 ; ac_reg_0 ; reset register
2559 0001421B 66EF <1> out dx, ax
2560 <1>
2561 <1> ; 06/06/2024
2562 <1> ; 01/06/2024
2563 <1> ;;;
2564 <1> ;call delay1_4ms
2565 <1> ;call delay1_4ms
2566 <1> ;call delay1_4ms
2567 <1> ;call delay1_4ms
2568 <1> ;;;
2569 <1>
2570 <1> ; 01/06/2024
2571 <1> ; 26/05/2024
2572 <1> ; 19/11/2023
2573 <1> ;call delay_100ms
2574 <1> ; 24/11/2023 - temporary
2575 <1> ;call delay_100ms
2576 <1> ;call delay_100ms
2577 <1> ;call delay_100ms
2578 <1>
2579 <1> ; 01/06/2024
2580 <1> ;;;
2581 0001421D 09ED <1> or ebp, ebp ; 21/11/2023
2582 0001421F 753E <1> jnz short _ac97_codec_init_ok
2583 <1> ;;;
2584 <1>
2585 00014221 31C0 <1> xor eax, eax ; 0
2586 00014223 668B15[D4880100] <1> mov dx, [NAMBAR]
2587 0001422A 6683C226 <1> add dx, CODEC_REG_POWERDOWN
2588 0001422E 66EF <1> out dx, ax
2589 <1>
2590 <1> ; 10/06/2017
2591 <1> ; 29/05/2017
2592 <1> ; wait for 1 second
2593 <1> ; 01/06/2024
2594 <1> ; 16/05/2024
2595 00014230 B9E8030000 <1> mov ecx, 1000 ; 1000*0.25ms
2596 <1> ; 20/11/2023
2597 <1> ;mov ecx, 10
2598 <1> ; 23/05/2024

```

```

2599      <1>      ;mov     cl, 10
2600      <1>      ; 24/11/2023 - temporary
2601      <1>      ;mov     cl, 40
2602      <1>      ; ecx = 40 ; 23/05/2024
2603      <1>      _ac97_codec_rloop:
2604      <1>      ; 01/06/2024
2605 00014235 E834F9FFFF      <1>      call    delay1_4ms
2606      <1>      ; 06/06/2024
2607 0001423A E82FF9FFFF      <1>      call    delay1_4ms
2608 0001423F E82AF9FFFF      <1>      call    delay1_4ms
2609 00014244 E825F9FFFF      <1>      call    delay1_4ms
2610      <1>      ; 01/06/2024
2611      <1>      ; 19/11/2023
2612      <1>      ; 22/11/2023
2613      <1>      ;push    ecx
2614      <1>      ;call    delay_100ms
2615      <1>      ;pop     ecx
2616      <1>
2617      <1>      ; 01/06/2024
2618      <1>      ;;;
2619 00014249 668B15[D4880100] <1>      mov     dx, [NAMBAR]
2620 00014250 6683C226      <1>      add     dx, CODEC_REG_POWERDOWN
2621      <1>      ;;;
2622 00014254 66ED      <1>      in      ax, dx
2623      <1>
2624      <1>      ; 06/06/2024
2625      <1>      ;call    delay1_4ms
2626      <1>
2627      <1>      ;and     ax, 0Fh
2628      <1>      ; 21/11/2023
2629 00014256 240F      <1>      and     al, 0Fh
2630 00014258 3C0F      <1>      cmp     al, 0Fh
2631 0001425A 7403      <1>      je      short _ac97_codec_init_ok
2632      <1>      ; 24/11/2023 - temporary
2633      <1>      ;je      short _ac97_codec_init_ok_
2634      <1>      _ac97_codec_yloop:
2635 0001425C E2D7      <1>      loop    _ac97_codec_rloop
2636      <1>      ; 22/11/2023
2637      <1>      ; cf = 1
2638      <1>      init_ac97_codec_err1:
2639      <1>      ;stc
2640      <1>      init_ac97_codec_err2:
2641 0001425E C3      <1>      retn
2642      <1>
2643      <1>      _ac97_codec_init_ok_:
2644      <1>      ;mov     dx, GLOB_STS_REG ; 30h
2645      <1>      ;add     dx, [NABMBAR]
2646      <1>      ;in      eax, dx
2647      <1>      ;test    eax, CTRL_ST_CREADY
2648      <1>      ;jnz    short _ac97_codec_init_ok
2649      <1>      ;stc
2650      <1>      ;retn
2651      <1>
2652      <1>      _ac97_codec_init_ok:
2653      <1>      ; 06/06/2024 (temporary)
2654      <1>      ; (this may not be needed)
2655      <1>      ;;;
2656      <1>      ; 24/05/2024
2657      <1>      ; 26/11/2023
2658      <1>      ; 23/11/2023
2659 0001425F 31C0      <1>      xor     eax, eax
2660      <1>      ; 21/11/2023 - temporary
2661      <1>      ; 19/11/2023
2662      <1>      mov     al, 2 ; force set 16-bit 2-channel PCM
2663 00014263 66BA2C00      <1>      mov     dx, GLOB_CNT_REG ; 2Ch
2664 00014267 660315[D6880100] <1>      add     dx, [NABMBAR]
2665 0001426E EF      <1>      out     dx, eax
2666      <1>      ;
2667 0001426F E8FAF8FFFF      <1>      call    delay1_4ms
2668      <1>      ;;;
2669      <1>
2670      <1>      ; 23/11/2023 - temporary
2671      <1>      ;call    delay_100ms
2672      <1>      ; 06/06/2024
2673      <1>      ;call    delay1_4ms
2674      <1>
2675      <1>      ; 10/06/2017
2676 00014274 E884020000      <1>      call    reset_ac97_controller
2677      <1>
2678      <1>      ; 21/11/2023
2679      <1>      ;call    delay1_4ms
2680      <1>
2681      <1>      ; 01/06/2024
2682      <1>      ; 24/05/2024
2683      <1>      ; 23/05/2024
2684      <1>      ; 21/11/2023 - temporary
2685 00014279 E8E3F8FFFF      <1>      call    delay_100ms ; 06/06/2024
2686      <1>
2687      <1>      ; call    setup_ac97_codec
2688      <1>      ;
2689      <1>      ;detect_ac97_codec:
2690      <1>      ; retn
2691      <1>
2692      <1>      setup_ac97_codec:
2693      <1>      ; 06/06/2024 - TRDOS 386 v2.0.8
2694      <1>      ;;;
2695 0001427E 668B15[D4880100] <1>      mov     dx, [NAMBAR]
2696 00014285 6683C22A      <1>      add     dx, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
2697 00014289 66ED      <1>      in      ax, dx
2698      <1>
2699      <1>      ;and     al, ~BIT1 ; Clear DRA
2700 0001428B 24FC      <1>      and     al, ~(BIT1+BIT0) ; Clear DRA+VRA
2701      <1>      ;or      al, BIT0 ; Set VRA
2702      <1>
2703 0001428D 668B15[D4880100] <1>      mov     dx, [NAMBAR]
2704 00014294 6683C22A      <1>      add     dx, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
2705 00014298 66EF      <1>      out     dx, ax
2706      <1>      ;;;
2707      <1>
2708      <1>      ; 24/11/2023
2709      <1>      ; 19/11/2023 - TRDOS 386 v2.0.7
2710 0001429A C605[08890100]01 <1>      mov     byte [VRA], 1
2711      <1>
2712      <1>      ; 25/11/2023 - temporary
2713      <1>      ;jmp     short vra_not_supported
2714      <1>
2715      <1>      ; 23/05/2024
2716      <1>      ; 18/05/2024
2717      <1>      ;call    delay1_4ms ; 06/06/2024
2718      <1>      ;call    delay_100ms
2719      <1>
2720      <1>      ; 06/06/2024
2721      <1>      ; 01/06/2024
2722      <1>      %if 0

```

```

2723 <1> ; 24/05/2024
2724 <1> ; 23/05/2024
2725 <1> mov dx, [NAMBAR]
2726 <1> add dx, CODEC_EXT_AUDIO_REG ; 28h
2727 <1> in ax, dx
2728 <1>
2729 <1> ; 18/05/2024
2730 <1> ; 02/12/2023
2731 <1> call delay1_4ms
2732 <1> ; 17/05/2024
2733 <1> ; call delay_100ms
2734 <1> ;
2735 <1> test al, 1 ; BIT0 ; Variable Rate Audio bit
2736 <1> jz short vra_not_supported
2737 <1>
2738 <1> mov dx, [NAMBAR]
2739 <1> add dx, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
2740 <1> in ax, dx
2741 <1>
2742 <1> ; 06/06/2024
2743 <1> ; 23/05/2024
2744 <1> ; 02/12/2023
2745 <1> ; call delay1_4ms
2746 <1> ; call delay_100ms ; 18/05/2024
2747 <1> ; 24/05/2024
2748 <1> ; call delay1_4ms
2749 <1> ; call delay1_4ms
2750 <1> ; call delay1_4ms
2751 <1>
2752 <1> ; and al, ~BIT1 ; Clear DRA
2753 <1> ;;;
2754 <1> ; 01/06/2024
2755 <1> and al, ~(BIT1+BIT0) ; Clear DRA+VRA
2756 <1> ;;;
2757 <1> ; 04/06/2024
2758 <1> ; mov dx, [NAMBAR]
2759 <1> ; add dx, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
2760 <1> ;;;
2761 <1> out dx, ax
2762 <1> %endif
2763 <1>
2764 <1> ; 04/06/2024
2765 <1> call delay1_4ms
2766 <1>
2767 <1> ; 01/06/2024
2768 <1> ; check VRA
2769 <1> mov dx, [NAMBAR]
2770 <1> add dx, CODEC_EXT_AUDIO_REG ; 28h
2771 <1> in ax, dx
2772 <1>
2773 <1> ; 06/06/2024
2774 <1> ; call delay1_4ms
2775 <1>
2776 <1> test al, 1 ; BIT0 ; Variable Rate Audio bit
2777 <1> jz short vra_not_supported
2778 <1>
2779 <1> inc ax ; 0FFh -> 0
2780 <1> jnz short not_alc ; skip ALC850 (and ALC655) checking
2781 <1> ; ax = -1 -> 0
2782 <1>
2783 <1> ;;;
2784 <1> ; 06/06/2024 (temporary!?)
2785 <1> ; check ALC850
2786 <1> ; (which does not support VRA but it may not be detected)
2787 <1> ;
2788 <1> call delay1_4ms
2789 <1> ;
2790 <1> mov dx, [NAMBAR]
2791 <1> add dx, CODEC_VENDOR_ID1 ; 7Ch
2792 <1> in ax, dx
2793 <1> shl eax, 16 ; *
2794 <1> ; cmp ax, 414Ch ; VENDOR_ID - 'AL'
2795 <1> ; jne short not_alc
2796 <1> mov dx, [NAMBAR]
2797 <1> add dx, CODEC_VENDOR_ID2 ; 7Eh
2798 <1> in ax, dx
2799 <1> ; inc edx
2800 <1> ; inc edx
2801 <1> ; in ax, dx ; VENDOR ID2
2802 <1>
2803 <1> inc eax ; 0FFFFFFFFh -> 0 ; 0 -> 1
2804 <1> jz short vra_not_supported
2805 <1> dec eax ; 1 -> 0
2806 <1> jz short vra_not_supported
2807 <1>
2808 <1> ror eax, 16
2809 <1> cmp ax, 414Ch ; VENDOR_ID - 'AL'
2810 <1> jne short not_alc
2811 <1> rol eax, 16
2812 <1>
2813 <1> ; ALC850
2814 <1> cmp ah, 47h ; 'g'
2815 <1> jne short not_alc
2816 <1> cmp al, 90h
2817 <1> ; cmp ax, 4790h ; VENDOR ID - 'G' ; bit 8-15
2818 <1> ; ; CHIP ID - 1001b ; bit 4-7
2819 <1> ; ; Version Number - 0 ; bit 0-3
2820 <1> ; jne short not_alc850 ; not_alc
2821 <1> je short vra_not_supported
2822 <1> ;
2823 <1> ; 06/06/2024
2824 <1> %if 0
2825 <1> not_alc850:
2826 <1> ; ALC655
2827 <1> cmp al, 60h
2828 <1> ; cmp ax, 4760h ; VENDOR ID - 'G' ; bit 8-15
2829 <1> ; ; CHIP ID - 0110b ; bit 4-7
2830 <1> ; ; Version Number - 0 ; bit 0-3
2831 <1> ; jne short not_alc655
2832 <1> je short vra_not_supported
2833 <1> ;
2834 <1> not_alc655:
2835 <1> %endif
2836 <1> not_alc:
2837 <1> ; 06/06/2024
2838 <1>
2839 <1> mov dx, [NAMBAR]
2840 <1> add dx, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
2841 <1> in ax, dx
2842 <1>
2843 <1> ; and al, ~BIT1 ; Clear DRA
2844 <1>
2845 <1> or al, AC97_EA_VRA ; 1 ; 04/11/2023
2846 <1>

```

```

2847      <1>      ; 06/06/2024
2848      <1>      ; 04/06/2024
2849      <1>      ;mov     dx, CODEC_EXT_AUDIO_CTRL_REG      ; 2Ah
2850      <1>      ;add     dx, [NAMBAR]
2851      <1>
2852      00014307 66EF      <1>      out     dx, ax          ; Enable variable rate audio
2853      <1>
2854      <1>      ; 01/06/2024
2855      00014309 B90A000000 <1>      mov     ecx, 10
2856      <1>      ; 21/11/2023
2857      <1>      ;mov     cl, 10
2858      <1>      check_vra:
2859      <1>      ; 01/06/2024
2860      <1>      ;call    delay_100ms
2861      0001430E E85BF8FFFF <1>      call    delay1_4ms
2862      <1>
2863      <1>      ;;
2864      <1>      ; 04/06/2024
2865      00014313 668B15[D4880100] <1>      mov     dx, [NAMBAR]
2866      0001431A 6683C22A <1>      add     dx, CODEC_EXT_AUDIO_CTRL_REG      ; 2Ah
2867      <1>      ;;
2868      <1>
2869      0001431E 66ED      <1>      in      ax, dx
2870      <1>
2871      <1>      ; 24/05/2024
2872      <1>      ; 18/05/2024
2873      <1>      ; 02/12/2023
2874      <1>      ;call    delay1_4ms
2875      <1>
2876      <1>      ;test    al, AC97_EA_VRA ; 1
2877      <1>      ; 04/06/2024
2878      00014320 2401      <1>      and     al, AC97_EA_VRA ; 1
2879      00014322 7508      <1>      jnz     short vra_ok
2880      <1>
2881      00014324 E2E8      <1>      loop    check_vra
2882      <1>
2883      <1>      vra_not_supported: ; 24/11/2023
2884      <1>      ; VRA is not usable
2885      <1>      ;mov     byte [VRA], 0
2886      00014326 FE0D[08890100] <1>      dec     byte [VRA]
2887      <1>
2888      <1>      ; 03/06/2024
2889      <1>      %if 0
2890      <1>      ; 02/06/2024
2891      <1>      jmp     short set_volume
2892      <1>
2893      <1>      ; 02/06/2024
2894      <1>      set_sampling_rate:
2895      <1>      mov     dx, [NAMBAR]
2896      <1>      add     dx, CODEC_PCM_FRONT_DACRATE_REG
2897      <1>      out     dx, ax
2898      <1>      retn
2899      <1>
2900      <1>      ; 02/06/2024
2901      <1>      get_sampling_rate:
2902      <1>      mov     dx, CODEC_PCM_FRONT_DACRATE_REG
2903      <1>      add     dx, [NAMBAR]
2904      <1>      in      ax, dx
2905      <1>      retn
2906      <1>      %endif
2907      <1>
2908      <1>      vra_ok:
2909      <1>
2910      <1>      ; 03/06/2024
2911      <1>      %if 0
2912      <1>      ; 02/06/2024
2913      <1>      ; a second test for verifying VRA status
2914      <1>      ; (may be needed for ALC850 codec and CK804 controller)
2915      <1>
2916      <1>      call    get_sampling_rate
2917      <1>      or      ax, ax
2918      <1>      jz      short vra_not_supported
2919      <1>      cmp     ax, 48000
2920      <1>      jne     short set_volume
2921      <1>      mov     ax, 24000
2922      <1>      call    set_sampling_rate
2923      <1>      call    get_sampling_rate
2924      <1>      cmp     ax, 48000
2925      <1>      je      short vra_not_supported
2926      <1>      mov     ax, 48000
2927      <1>      call    set_sampling_rate
2928      <1>      ;call    get_sampling_rate
2929      <1>      %endif
2930      <1>
2931      <1>      ; 02/06/2024
2932      <1>      set_volume:
2933      <1>      ; 24/05/2024
2934      <1>      ; 18/05/2024 - TRDOS 386 v2.0.8
2935      <1>      ; 20/11/2023
2936      <1>      ; 19/11/2023 - TRDOS 386 v2.0.7
2937      <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
2938      <1>      ; 22/07/2020
2939      <1>      ; 10/06/2017
2940      <1>      ; 29/05/2017
2941      <1>
2942      <1>      ;mov     eax, 0202h
2943      <1>      ; 21/11/2023
2944      <1>      ;mov     eax, 0B0Bh
2945      <1>      ; 23/11/2023
2946      0001432C B81D1D0000 <1>      mov     eax, 1D1Dh
2947      00014331 66A3[0C890100] <1>      mov     [audio_master_volume], ax
2948      <1>      ; 24/05/2024
2949      00014337 66A3[0E890100] <1>      mov     [audio_pcmo_volume], ax
2950      <1>      ;mov     ax, 1F1Fh ; 31, 31
2951      <1>
2952      <1>      ; 21/11/2023
2953      <1>      ;push    eax
2954      <1>
2955      <1>      ; 20/11/2023
2956      <1>      ;mov     ax, 0B0Bh
2957      <1>      ; 23/11/2023
2958      0001433D 66B80202 <1>      mov     ax, 0202h
2959      <1>
2960      00014341 668B15[D4880100] <1>      mov     dx, [NAMBAR]
2961      00014348 6683C202 <1>      add     dx, CODEC_MASTER_VOL_REG      ;02h
2962      <1>      ;xor     ax, ax ; volume attenuation = 0 (max. volume)
2963      <1>      ; 06/08/2022
2964      <1>      ;xor     eax, eax
2965      <1>      ; 19/11/2023
2966      0001434C 66EF      <1>      out     dx, ax
2967      <1>
2968      <1>      ; 01/06/2024
2969      <1>      ; 23/05/2024
2970      <1>      ; 16/05/2024

```

```

2971      <1>      ; 20/11/2023
2972 0001434E E81BF8FFFF <1>      call delay1_4ms
2973      <1>      ;call delay1_4ms
2974      <1>      ;call delay1_4ms
2975      <1>      ;call delay1_4ms
2976      <1>      ; 21/11/2023 temporary
2977      <1>      ;call delay_100ms ; 18/05/2024
2978      <1>
2979      <1>      ; 21/11/2023
2980      <1>      ;pop     eax
2981      <1>
2982      <1>      ; 23/11/2023
2983      <1>      ; 21/11/2023
2984      <1>      ;mov     ax, 0202h
2985      <1>
2986 00014353 668B15[D4880100] <1>      mov     dx, [NAMBAR]
2987 0001435A 6683C218 <1>      add     dx, CODEC_PCM_OUT_REG ;18h
2988      <1>      ;xor     ax, ax
2989 0001435E 66EF <1>      out     dx, ax
2990      <1>
2991      <1>      ; 01/06/2024
2992      <1>      ; 23/05/2024
2993      <1>      ; 16/05/2024
2994      <1>      ; 20/11/2023
2995      <1>      ;call     delay1_4ms
2996      <1>      ;call     delay1_4ms
2997      <1>      ;call     delay1_4ms
2998      <1>      ;call     delay1_4ms
2999      <1>      ; 21/11/2023 - temporary
3000      <1>      ;call     delay_100ms ; 18/05/2024
3001      <1>
3002      <1>      ; 23/11/2023
3003      <1>      %if 0
3004      <1>      mov     dx, [NAMBAR]
3005      <1>      add     dx, CODEC_MASTER_MONO_VOL_REG ;06h
3006      <1>      ;xor     ax, ax
3007      <1>      out     dx, ax
3008      <1>
3009      <1>      ;call     delay1_4ms
3010      <1>      ;call     delay1_4ms
3011      <1>      ;call     delay1_4ms
3012      <1>      ;call     delay1_4ms
3013      <1>
3014      <1>      mov     dx, [NAMBAR]
3015      <1>      add     dx, CODEC_PCBEPP_VOL_REG ;0Ah
3016      <1>      ;xor     ax, ax
3017      <1>      out     dx, ax
3018      <1>
3019      <1>      ;call     delay1_4ms
3020      <1>      ;call     delay1_4ms
3021      <1>      ;call     delay1_4ms
3022      <1>      ;call     delay1_4ms
3023      <1>
3024      <1>      mov     ax, 8008h ; Mute
3025      <1>      mov     dx, [NAMBAR]
3026      <1>      ; 22/07/2020
3027      <1>      add     dx, CODEC_PHONE_VOL_REG ;0Ch
3028      <1>      ; AC97_PHONE_VOL ; TAD Input (Mono)
3029      <1>      out     dx, ax
3030      <1>
3031      <1>      ;call     delay1_4ms
3032      <1>      ;call     delay1_4ms
3033      <1>      ;call     delay1_4ms
3034      <1>      ;call     delay1_4ms
3035      <1>
3036      <1>      mov     ax, 0808h
3037      <1>      mov     dx, [NAMBAR]
3038      <1>      add     dx, CODEC_LINE_IN_VOL_REG ;10h ; Line Input (Stereo)
3039      <1>      out     dx, ax
3040      <1>
3041      <1>      ;call     delay1_4ms
3042      <1>      ;call     delay1_4ms
3043      <1>      ;call     delay1_4ms
3044      <1>      ;call     delay1_4ms
3045      <1>
3046      <1>      ;mov     ax, 0808h
3047      <1>      mov     dx, [NAMBAR]
3048      <1>      add     dx, CODEC_CD_VOL_REG ;12h ; CR Input (Stereo)
3049      <1>      out     dx, ax
3050      <1>
3051      <1>      ;call     delay1_4ms
3052      <1>      ;call     delay1_4ms
3053      <1>      ;call     delay1_4ms
3054      <1>      ;call     delay1_4ms
3055      <1>
3056      <1>      ;mov     ax, 0808h
3057      <1>      mov     dx, [NAMBAR]
3058      <1>      add     dx, CODEC_AUX_VOL_REG ;16h ; Aux Input (Stereo)
3059      <1>      out     dx, ax
3060      <1>
3061      <1>      ;call     delay1_4ms
3062      <1>      ;call     delay1_4ms
3063      <1>      ;call     delay1_4ms
3064      <1>      ;call     delay1_4ms
3065      <1>
3066      <1>      ; 21/11/2023 - temporary
3067      <1>      ;call     delay_100ms
3068      <1>      %endif
3069      <1>      ; 16/05/2024
3070      <1>      ;clc
3071      <1>
3072      <1>      set_volume_ok:
3073      <1>
3074      <1>      ;detect_ac97_codec:
3075 00014360 C3 <1>      retn
3076      <1>
3077      <1>      set_ac97_bdl: ; Set AC97 (ICH) Buffer Descriptor List
3078      <1>      ; 04/06/2024 - TRDOS 386 v2.0.8
3079      <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
3080      <1>      ; 17/06/2017
3081      <1>      ; 11/06/2017
3082      <1>      ; 28/05/2017
3083      <1>      ; eax = dma buffer address = [audio_DMA_buff]
3084      <1>      ; ; ecx = dma buffer buffer size = [audio_dmabuff_size]
3085      <1>
3086      <1>      ; 04/06/2024
3087      <1>      ; ecx = DMA half buffer size as truncated for 8 byte alignment
3088      <1>
3089      <1>      ;shr     ecx, 1 ; dma half buffer size
3090 00014361 89CE <1>      mov     esi, ecx
3091      <1>
3092 00014363 BF[10890100] <1>      mov     edi, audio_bdl_buff ; get BDL address
3093      <1>      ;mov     ecx, 32 / 2 ; make 32 entries in BDL
3094      <1>      ; 06/08/2022

```

```

3095 00014368 29C9      <1>      sub     ecx, ecx
3096 0001436A B110      <1>      mov     cl, 16
3097                                <1>
3098 0001436C EB05      <1>      jmp     short s_ac97_bd11
3099                                <1>
3100                                <1> s_ac97_bd10:
3101                                <1>      ; set buffer descriptor 0 to start of data file in memory
3102                                <1>
3103 0001436E A1[F0880100] <1>      mov     eax, [audio_dma_buff] ; Physical address of DMA buffer
3104                                <1>
3105                                <1> s_ac97_bd11:
3106 00014373 AB          <1>      stosd                      ; store dmabuffer1 address
3107                                <1>
3108 00014374 89C2      <1>      mov     edx, eax
3109                                <1>
3110                                <1> ;
3111                                <1> ; Buffer Descriptors List
3112                                <1> ; As stated earlier, each buffer descriptor list is a set of (up to) 32
3113                                <1> ; descriptors, each 8 bytes in length. Bytes 0-3 of a descriptor entry point
3114                                <1> ; to a chunk of memory to either play from or record to. Bytes 4-7 of an
3115                                <1> ; entry describe various control things detailed below.
3116                                <1> ;
3117                                <1> ; Buffer pointers must always be aligned on a Dword boundry.
3118                                <1> ;
3119                                <1> ;
3120                                <1> ;IOCequ      BIT31          ; Fire an interrupt whenever this
3121                                <1> ; buffer is complete.
3122                                <1> ;
3123                                <1> ;BUPequ      BIT30          ; Buffer Underrun Policy.
3124                                <1> ; if this buffer is the last buffer
3125                                <1> ; in a playback, fill the remaining
3126                                <1> ; samples with 0 (silence) or not.
3127                                <1> ; It's a good idea to set this to 1
3128                                <1> ; for the last buffer in playback,
3129                                <1> ; otherwise you're likely to get a lot
3130                                <1> ; of noise at the end of the sound.
3131                                <1> ;
3132                                <1> ;
3133                                <1> ; Bits 15:0 contain the length of the buffer, in number of samples, which
3134                                <1> ; are 16 bits each, coupled in left and right pairs, or 32bits each.
3135                                <1> ; Luckily for us, that's the same format as .wav files.
3136                                <1> ;
3137                                <1> ; A value of FFFF is 65536 samples. Running at 44.1khz, that's just about
3138                                <1> ; 1.5 seconds of sample time. FFFF * 32bits is 1FFFFh bytes or 128k of data.
3139                                <1> ;
3140                                <1> ; A value of 0 in these bits means play no samples.
3141                                <1> ;
3142                                <1> ;
3143 00014376 89F0      <1>      mov     eax, esi ; DMA half buffer size
3144                                <1>
3145                                <1> ; 04/06/2024 - Erdogan Tan
3146                                <1> ; NOTE: I have changed DMA half buffer size to truncated
3147                                <1> ; (8 byte aligned) audio buffer size for smooth audio playing
3148                                <1> ; (it was page border aligned before June 4, 2024)
3149                                <1> ;
3150                                <1> ; For example:
3151                                <1> ;   For 65416 bytes audio buffer (22kHz, 16bit samples)
3152                                <1> ;   (user's audio buff virtual addr is mapped to physical)
3153                                <1> ;   ((kernel's and user's audio buff pages are same))
3154                                <1> ;   1) Memory allocation (of it) is 65536 bytes
3155                                <1> ;   2) DMA half buffer size is 65416 bytes
3156                                <1> ;   (it would be 65536 bytes before this modification)
3157                                <1> ;   ((additional 140 bytes would cause to a noise))
3158                                <1> ;   3) Total DMA buffer size is 131072 bytes
3159                                <1> ;   (the last 240 bytes will not be used for playing)
3160                                <1> ;
3161                                <1> ; (buffer will be truncated if the size is not a multiple of 8)
3162                                <1> ;
3163 00014378 01C2      <1>      add     edx, eax
3164 0001437A D1E8      <1>      shr     eax, 1 ; count of 16 bit samples
3165                                <1> ; 19/11/2023
3166 0001437C 0D000000C0 <1>      or      eax, IOC+BUSP
3167                                <1> ;or      eax, IOC ; 11/06/2017
3168 00014381 AB          <1>      stosd
3169                                <1> ;
3170                                <1> ; 2nd buffer:
3171                                <1> ;
3172 00014382 89D0      <1>      move    eax, edx ; Physical address of the 2nd half of DMA buffer
3173 00014384 AB          <1>      stosd                      ; store dmabuffer2 address
3174                                <1> ;
3175                                <1> ; set length to [audio_dmabuff_size]/2
3176                                <1> ; Set control (bits 31:16) to BUP, bits 15:0=number of samples
3177                                <1> ;
3178 00014385 89F0      <1>      mov     eax, esi ; DMA half buffer size
3179 00014387 D1E8      <1>      shr     eax, 1 ; count of 16 bit samples
3180                                <1> ; 19/11/2023
3181 00014389 0D000000C0 <1>      or      eax, IOC+BUSP
3182                                <1> ;or      eax, IOC ; 11/06/2017
3183 0001438E AB          <1>      stosd
3184                                <1> ;
3185 0001438F E2DD      <1>      loop     s_ac97_bd10
3186                                <1> ;
3187 00014391 C3          <1>      retn
3188                                <1> ;
3189                                <1> ac97_start_play:
3190                                <1> ; 26/05/2024 - TRDOS 386 v2.0.8
3191                                <1> ; 26/11/2023
3192                                <1> ; 20/11/2023
3193                                <1> ; 19/11/2023 - TRDOS 386 v2.0.7
3194                                <1> ; 06/08/2022 - TRDOS 386 v2.0.5
3195                                <1> ; 28/05/2017
3196                                <1> ; Derived from 'playwav' procedure in 'ICHWAV.ASM'
3197                                <1> ; .wav player for DOS by Jeff Leyda (02/09/2002)
3198                                <1> ;
3199                                <1> ; set output rate
3200                                <1> ; entry: [audio_freq] = desired sample rate
3201                                <1> ;
3202                                <1> ; 21/11/2023 - temporary
3203                                <1> ; call ac97_codec_config
3204                                <1> ;
3205                                <1> ; 20/11/2023
3206                                <1> %if 0
3207                                <1> AC97_EA_VRA equ 0001h ; 04/11/2023
3208                                <1> ;
3209                                <1> mov     dx, [NAMBAR]
3210                                <1> add     dx, CODEC_EXT_AUDIO_CTRL_REG ; 2Ah
3211                                <1> in      ax, dx
3212                                <1> ;
3213                                <1> ; 23/05/2024
3214                                <1> call    delay1_4ms
3215                                <1> ;
3216                                <1> and     al, ~BIT1 ; Clear DRA
3217                                <1> or      al, AC97_EA_VRA ; 1 ; 04/11/2023
3218                                <1> out     dx, ax ; Enable variable rate audio

```

```

3219 <1>
3220 <1> ;call delay1_4ms
3221 <1> ;call delay1_4ms
3222 <1> ;call delay1_4ms
3223 <1> ;call delay1_4ms
3224 <1>
3225 <1> ; 23/05/2024
3226 <1> call delay_100ms
3227 <1> %endif
3228 <1>
3229 <1> ; 26/11/2023
3230 <1> ; 24/11/2023 - temporary
3231 <1> ;mov ax, 48000
3232 <1>
3233 <1> ; 20/11/2023
3234 00014392 803D[08890100]01 <1> cmp byte [VRA], 1
3235 00014399 7213 <1> jb short skip_set_rate
3236 <1>
3237 0001439B 66A1[06890100] <1> mov ax, [audio_freq] ; sample rate
3238 <1>
3239 <1> ;skip_set_rate: ; 24/11/2023
3240 000143A1 668B15[D4880100] <1> mov dx, [NABMBAR]
3241 000143A8 6683C22C <1> add dx, CODEC_PCM_FRONT_DACRATE_REG ; 2Ch
3242 000143AC 66EF <1> out dx, ax ; out sample rate
3243 <1>
3244 <1> ; 01/06/2024
3245 <1> ; 27/11/2023
3246 <1> ;call delay1_4ms
3247 <1> ;call delay1_4ms
3248 <1> ;call delay1_4ms
3249 <1> ;call delay1_4ms
3250 <1> ; 27/11/2023 - temporary
3251 <1> ;call delay_100ms
3252 <1>
3253 <1> ; 26/11/2023
3254 <1> skip_set_rate: ; 20/11/2023
3255 <1>
3256 <1> ; 23/11/2023 - temporary
3257 <1> ;push ebx
3258 <1> ;mov ebx, 0B8000h
3259 <1> ;mov al, 'x'
3260 <1> ;mov ah, 4Eh
3261 <1> ;mov [ebx], ax
3262 <1> ;pop ebx
3263 <1> ;
3264 <1> ; register reset the DMA engine. This may cause a pop noise on the output
3265 <1> ; lines when the device is reset. Prolly a better idea to mute output, then
3266 <1> ; reset.
3267 <1> ;
3268 <1> ; 21/11/2023 - temporary
3269 <1> ; 20/11/2023
3270 <1> ;mov dx, [NABMBAR]
3271 <1> ;add dx, PO_CR_REG ; set pointer to Cntl reg
3272 <1> ;mov al, RR ; set reset
3273 <1> ;out dx, al ; self clearing bit
3274 <1>
3275 <1> ; 23/11/2023 - temporary
3276 <1> ;call delay_100ms
3277 <1>
3278 <1> ; mov edi, audio_bdl_buff
3279 <1> ; mov edx, [audio_dmabuff_size]
3280 <1> ; shr edx, 1
3281 <1> ; mov ecx, 32/2
3282 <1> ;ac97_set_bdl_buffer:
3283 <1> ; ; 1st half of DMA buffer
3284 <1> ; mov eax, [audio_dma_buff]
3285 <1> ; push eax
3286 <1> ; stosd
3287 <1> ; mov eax, edx ; dma buffer size / 2
3288 <1> ; or eax, IOC+BUS
3289 <1> ; stosd
3290 <1> ; pop eax
3291 <1> ; ; 2nd half of DMA buffer
3292 <1> ; add eax, edx
3293 <1> ; stosd
3294 <1> ; mov eax, edx ; dma buffer size / 2
3295 <1> ; or eax, IOC+BUS
3296 <1> ; stosd
3297 <1> ; loop ac97_set_bdl_buffer
3298 <1>
3299 <1> ; tell the DMA engine where to find our list of Buffer Descriptors.
3300 <1> ; this 32bit value is a flat mode memory offset (ie no segment:offset)
3301 <1>
3302 <1> ; write NABMBAR+10h with offset of buffer descriptor list
3303 <1> ;
3304 000143AE 88[10890100] <1> ; mov eax, audio_bdl_buff
3305 000143B3 668B15[D6880100] <1> mov dx, [NABMBAR]
3306 000143BA 6683C210 <1> add dx, PO_BDBAR_REG
3307 000143BE EF <1> out dx, eax
3308 <1>
3309 <1> ; 01/06/2024
3310 <1> ; 24/05/2024
3311 <1> ; 23/11/2023 - temporary
3312 <1> ;call delay_100ms
3313 <1> ; 23/05/2024
3314 <1> ;call delay1_4ms
3315 <1> ;
3316 <1> ; All set. Let's play some music.
3317 <1> ;
3318 <1> ;mov eax, 31
3319 <1> ; 06/08/2022
3320 000143BF 29C0 <1> sub eax, eax
3321 000143C1 B01F <1> mov al, 31
3322 000143C3 E846000000 <1> call set_ac97_LastValidIndex
3323 <1>
3324 <1> ; 01/06/2024
3325 <1> ; 24/05/2024
3326 <1> ; 23/11/2023 - temporary
3327 <1> ;call delay_100ms
3328 <1> ; 23/05/2024
3329 <1> ;call delay1_4ms
3330 <1> ; 27/05/2024
3331 <1> ;call delay1_4ms
3332 <1> ;call delay1_4ms
3333 <1> ;call delay1_4ms
3334 <1>
3335 000143C8 C605[0A890100]01 <1> mov byte [audio_play_cmd], 1 ; play command (do not stop) !
3336 <1>
3337 <1> ac97_play: ; continue to play (after pause)
3338 <1> ; 27/05/2024
3339 <1> ;
3340 <1> ; 19/05/2024 - temporary
3341 <1> ; 24/11/2023 - temporary
3342 <1> ;mov ax, 0202h

```

```

3343                                     <1>      ; 23/05/2024
3344 000143CF 66B81F1F                 <1>      mov     ax, 1F1Fh
3345 000143D3 662B05[0C890100]         <1>      sub     ax, [audio_master_volume]
3346                                     <1>
3347 000143DA 66B815[D4880100]         <1>      mov     dx, [NAMBAR]
3348 000143E1 6683C202                 <1>      add     dx, CODEC_MASTER_VOL_REG ; 02h ; Line Out
3349 000143E5 66EF                     <1>      out     dx, ax
3350                                     <1>
3351                                     <1>      ; 01/06/2024
3352                                     <1>      ; 19/05/2024 - temporary
3353                                     <1>      ;call delay_100ms
3354                                     <1>      ; 23/05/2024
3355                                     <1>      ;call delay1_4ms
3356                                     <1>      ; 24/05/2024
3357                                     <1>      ;call delay1_4ms
3358                                     <1>      ;call delay1_4ms
3359                                     <1>      ;call delay1_4ms
3360                                     <1>
3361                                     <1>      ;mov ax, 0202h
3362                                     <1>      ; 23/05/2024
3363                                     <1>      ;mov ax, 1F1Fh
3364                                     <1>      ;sub ax, [audio_master_volume]
3365                                     <1>
3366                                     <1>      ; 27/05/2024
3367 000143E7 66B81F1F                 <1>      mov     ax, 1F1Fh
3368 000143EB 662B05[0E890100]         <1>      sub     ax, [audio_pcmo_volume]
3369                                     <1>
3370 000143F2 66B815[D4880100]         <1>      mov     dx, [NAMBAR]
3371 000143F9 6683C218                 <1>      add     dx, CODEC_PCM_OUT_REG ; 18h ; PCM Out
3372 000143FD 66EF                     <1>      out     dx, ax
3373                                     <1>
3374                                     <1>      ; 01/06/2024
3375                                     <1>      ; 24/05/2024
3376                                     <1>      ; 23/05/2024
3377                                     <1>      ;call delay1_4ms
3378                                     <1>      ;call delay1_4ms
3379                                     <1>      ;call delay1_4ms
3380                                     <1>      ;call delay1_4ms
3381                                     <1>      ; 24/05/2024
3382                                     <1>      ;call delay_100ms
3383                                     <1>      ;;
3384                                     <1>
3385                                     <1>      ; 11/06/2017
3386                                     <1>      ; 29/05/2017
3387                                     <1>      ; 28/05/2017
3388 000143FF 66B815[D6880100]         <1>      mov     dx, [NABMBAR]
3389 00014406 6683C21B                 <1>      add     dx, PO_CR_REG ; PCM out control register
3390                                     <1>      ; 26/11/2023
3391 0001440A B011                     <1>      mov     al, IOCE+RPBM ; 29/05/2017
3392                                     <1>      ; 24/11/2023
3393                                     <1>      ;mov al, 1dh ; (Ref: KolibriOS, intelac97.asm, 'play:')
3394 0001440C EE                       <1>      out     dx, al ; set start!
3395                                     <1>
3396                                     <1>      ;mov byte [audio_play_cmd], 1 ; play command (do not stop) !
3397                                     <1>
3398 0001440D C3                       <1>      retn
3399                                     <1>
3400                                     <1>      ;input AL = index # to stop on
3401                                     <1>      set_ac97_LastValidIndex:
3402                                     <1>      ; 28/05/2017
3403                                     <1>      ; Derived from 'setLastValidIndex' procedure in 'ICHWAV.ASM'
3404                                     <1>      ; .wav player for DOS by Jeff Leyda (02/09/2002)
3405 0001440E 66B815[D6880100]         <1>      mov     dx, [NABMBAR]
3406 00014415 6683C215                 <1>      add     dx, PO_LVI_REG
3407 00014419 EE                       <1>      out     dx, al
3408                                     <1>      ;mov [audio_lvi], al ; for ac97_int_handler
3409 0001441A C3                       <1>      retn
3410                                     <1>
3411                                     <1>      ac97_volume:
3412                                     <1>      ; 23/05/2024
3413                                     <1>      ; 28/05/2017
3414                                     <1>      ; b1 = component (0 = master/playback/lineout volume)
3415                                     <1>      ; (1 = PCM Out volume) ; 23/05/2024
3416                                     <1>      ; c1 = left channel volume level (0 to 31)
3417                                     <1>      ; ch = right channel volume level (0 to 31)
3418                                     <1>
3419                                     <1>      ; 24/05/2024
3420 0001441B 80FB01                 <1>      cmp     b1, 1
3421 0001441E 7727                     <1>      ja     short ac97_vol_2 ; temporary !
3422                                     <1>
3423 00014420 66B81F1F                 <1>      mov     ax, 1F1Fh ; 31,31
3424 00014424 38C1                     <1>      cmp     c1, al
3425 00014426 771F                     <1>      ja     short ac97_vol_2
3426 00014428 38E5                     <1>      cmp     ch, ah
3427 0001442A 771B                     <1>      ja     short ac97_vol_2
3428                                     <1>
3429 0001442C 08DB                     <1>      or      b1, b1
3430 0001442E 7518                     <1>      jnz     short ac97_vol_1
3431                                     <1>
3432                                     <1>      ; 23/05/2024
3433 00014430 66BA0200                 <1>      mov     dx, CODEC_MASTER_VOL_REG ; 02h ; Line Out
3434                                     <1>      ; 24/05/2024
3435 00014434 66890D[0C890100]         <1>      mov     [audio_master_volume], cx
3436                                     <1>      ac97_vol_0:
3437 0001443B 6629C8                 <1>      sub     ax, cx
3438                                     <1>      ;mov dx, [NAMBAR]
3439                                     <1>      ;add dx, CODEC_MASTER_VOL_REG
3440                                     <1>      ; 23/05/2024
3441 0001443E 660315[D4880100]         <1>      add     dx, [NAMBAR]
3442 00014445 66EF                     <1>      out     dx, ax
3443                                     <1>      ; 23/05/2024
3444                                     <1>      ac97_vol_2:
3445                                     <1>      ; 21/11/2023
3446                                     <1>      ;_ac97_ih5: ; 06/08/2022
3447 00014447 C3                       <1>      retn
3448                                     <1>
3449                                     <1>      ac97_vol_1:
3450                                     <1>      ; 24/05/2024
3451                                     <1>      ; 23/05/2024
3452                                     <1>      ;cmp b1, 1
3453                                     <1>      ;ja short ac97_vol_2 ; temporary !
3454                                     <1>      ; BL = 1
3455                                     <1>      ; PCM OUT volume
3456 00014448 66BA1800                 <1>      mov     dx, CODEC_PCM_OUT_REG ; 18h ; PCM out
3457                                     <1>      ; 24/05/2024
3458 0001444C 66890D[0E890100]         <1>      mov     [audio_pcmo_volume], cx
3459 00014453 EBE6                     <1>      jmp     short ac97_vol_0
3460                                     <1>
3461                                     <1>      ac97_int_handler:
3462                                     <1>      ; 04/06/2024
3463                                     <1>      ; 03/06/2024
3464                                     <1>      ; 02/06/2024 - TRDOS 386 v2.0.8
3465                                     <1>      ; 27/11/2023
3466                                     <1>      ; 24/11/2023

```



```

3467      <1>      ; 21/11/2023
3468      <1>      ; 20/11/2023 - TRDOS 386 v2.0.7
3469      <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
3470      <1>      ; 12/10/2017
3471      <1>      ; 10/10/2017
3472      <1>      ; 09/10/2017
3473      <1>      ; 13/06/2017, 13/06/2017
3474      <1>      ; 10/06/2017, 11/06/2017
3475      <1>      ; Interrupt Handler for AC97 (ICH) Audio Controller
3476      <1>      ; Note: called by 'dev_IRQ_service'
3477      <1>      ; 28/05/2017
3478      <1>
3479      <1>      ;push  eax ; * must be saved !
3480      <1>      ;push  edx
3481      <1>      ;push  ecx
3482      <1>      ;push  ebx ; * must be saved !
3483      <1>      ;push  esi
3484      <1>      ;push  edi
3485      <1>
3486      <1>      ; 02/06/2024
3487      <1>      %if 0
3488      <1>
3489      <1>      ;cmp    byte [audio_busy], 1
3490      <1>      ;jnb   _ac97_ih2 ; busy !
3491      <1>
3492      <1>      ; 24/11/2023
3493      <1>      ;mov    dx, [NABMBAR]
3494      <1>      ;mov    dx, PO_SR_REG
3495      <1>      ;in     ax, dx
3496      <1>      ;test   al, BCIS ; bit 3, 8
3497      <1>      ;jz     short _ac97_ih5
3498      <1>
3499      <1>      ; 24/11/2023
3500      <1>      mov    dx, GLOB_STS_REG
3501      <1>      add    dx, [NABMBAR]
3502      <1>      in     eax, dx
3503      <1>
3504      <1>      cmp    eax, 0FFFFFFFh ; -1
3505      <1>      ;je     _ac97_ih3 ; exit
3506      <1>      ; 06/08/2022
3507      <1>      je     short _ac97_ih5
3508      <1>
3509      <1>      ;test   eax, 40h ; PCM Out Interrupt
3510      <1>      ; 06/08/2022
3511      <1>      test   al, 40h
3512      <1>      jnz    short _ac97_ih0
3513      <1>
3514      <1>      ; 24/11/2023
3515      <1>      test   eax, eax
3516      <1>      ;jz     _ac97_ih3 ; exit
3517      <1>      ; 06/08/2022
3518      <1>      jz     short _ac97_ih5
3519      <1>
3520      <1>      ;mov    dx, GLOB_STS_REG
3521      <1>      ;add    dx, [NABMBAR]
3522      <1>      out    dx, eax
3523      <1>
3524      <1>      ;jmp    _ac97_ih3 ; exit
3525      <1>
3526      <1>      ; 06/08/2022
3527      <1>      retn
3528      <1>      ; 21/11/2023
3529      <1>      ;jmp    short _ac97_ih5
3530      <1>
3531      <1>      _ac97_ih0:
3532      <1>      ; 24/11/2023
3533      <1>      ;push  eax
3534      <1>      ; 09/10/2017
3535      <1>      cmp    byte [audio_play_cmd], 1
3536      <1>      ;jb     short _ac97_ih4 ; stop command !
3537      <1>      ; 24/11/2023
3538      <1>      jnb    short _ac97_ih4
3539      <1>
3540      <1>      ;mov    dx, GLOB_STS_REG
3541      <1>      ;add    dx, [NABMBAR]
3542      <1>      out    dx, eax
3543      <1>      ;jmp    short _ac97_stop
3544      <1>
3545      <1>      ; 24/11/2023
3546      <1>      ac97_stop:
3547      <1>      ; 28/05/2017
3548      <1>      mov    byte [audio_play_cmd], 0 ; stop !
3549      <1>
3550      <1>      %else
3551      <1>      ; 02/06/2024
3552      <1>      _ac97_ih0:
3553      <1>      cmp    byte [audio_play_cmd], 1
3554      <1>      jnb    short _ac97_ih4
3555      <1>      %endif
3556      <1>
3557      <1>      _ac97_stop: ; 09/10/2017
3558      <1>      ; 29/05/2017
3559      <1>      ;mov    dx, [NABMBAR]
3560      <1>      ;add    dx, PO_CR_REG
3561      <1>      ;mov    al, 0
3562      <1>      ;out    dx, al
3563      <1>
3564      <1>      ; 11/06/2017
3565      <1>      xor    al, al ; 0
3566      <1>      call   ac97_po_cmd
3567      <1>
3568      <1>      ; (Ref: KolibriOS, intelac97.asm, 'stop:')
3569      <1>      ; Clear FIFOE, BCIS, LVBCI (Ref: Intel ICH hub manual)
3570      <1>      mov    ax, 1Ch
3571      <1>      mov    dx, [NABMBAR]
3572      <1>      add    dx, PO_SR_REG
3573      <1>      out    dx, ax
3574      <1>
3575      <1>      ;retn
3576      <1>
3577      <1>      ; 11/06/2017
3578      <1>      mov    al, RR
3579      <1>      ac97_po_cmd:
3580      <1>      ; 11/06/2017
3581      <1>      ; 29/05/2017
3582      <1>      mov    dx, [NABMBAR]
3583      <1>      add    dx, PO_CR_REG ; PCM out control register
3584      <1>      out    dx, al
3585      <1>      retn
3586      <1>
3587      <1>      ; 02/06/2024
3588      <1>      %if 1
3589      <1>      ; 24/11/2023
3590      <1>      ac97_stop:

```

```

3591                                     <1>      ; 28/05/2017
3592 00014485 C605[0A890100]00      <1>      mov     byte [audio_play_cmd], 0 ; stop !
3593 0001448C EBD0                  <1>      jmp      short _ac97_stop
3594                                     <1>      %endif
3595                                     <1>
3596                                     <1>      ac97_pause:
3597                                     <1>      ; 11/06/2017
3598                                     <1>      ; 29/05/2017
3599 0001448E B010                  <1>      mov     al, IOCE
3600 00014490 EBE6                  <1>      jmp      short ac97_po_cmd
3601                                     <1>
3602                                     <1>      _ac97_ih4:
3603                                     <1>      ;mov     byte [audio_busy], 1
3604                                     <1>
3605                                     <1>      ; 02/06/2024
3606                                     <1>      %if 1
3607                                     <1>      ; 24/11/2023 (TRDOS386 'audio.s')
3608 00014492 668B15[D6880100]      <1>      mov     dx, [NABMBAR]
3609 00014499 6683C216              <1>      add     dx, PO_SR_REG
3610 0001449D 66ED                  <1>      in      ax, dx
3611                                     <1>
3612 0001449F A808                  <1>      test    al, BCIS ; bit 3, 8
3613 000144A1 744C                  <1>      jz      short _ac97_ih2      ; 02/06/2024
3614                                     <1>
3615                                     <1>      ; 02/06/2024
3616 000144A3 50                    <1>      push    eax ; *
3617                                     <1>      %else
3618                                     <1>      ; 27/05/2024
3619                                     <1>      ; 24/11/2023
3620                                     <1>      push    eax
3621                                     <1>
3622                                     <1>      mov     ax, 1Ch ; FIFOE(=16)+BCIS(=8)+LVBCI(=4)
3623                                     <1>      mov     dx, PO_SR_REG
3624                                     <1>      add     dx, [NABMBAR]
3625                                     <1>      out     dx, ax
3626                                     <1>
3627                                     <1>      mov     dx, PO_CIV_REG
3628                                     <1>      add     dx, [NABMBAR]
3629                                     <1>      in      al, dx
3630                                     <1>
3631                                     <1>      ;cmp     al, [audio_civ] ; [audio_flag]
3632                                     <1>      ;je      short _ac97_ih2
3633                                     <1>
3634                                     <1>      ; 21/11/2023
3635                                     <1>      ;mov     [audio_civ], al
3636                                     <1>      ; 20/11/2023
3637                                     <1>      mov     ah, al
3638                                     <1>      dec     al
3639                                     <1>      ;inc     al ; 11/06/2017
3640                                     <1>      and     al, 1Fh
3641                                     <1>      %endif
3642                                     <1>      ; 02/06/2024
3643 000144A4 66BA1400              <1>      mov     dx, PO_CIV_REG
3644 000144A8 660315[D6880100]      <1>      add     dx, [NABMBAR]
3645                                     <1>      ;in      ax, dx
3646                                     <1>      ; 03/06/2024
3647 000144AF EC                    <1>      in      al, dx
3648                                     <1>      ; al = CVI, ah = LVI
3649                                     <1>      ;xchg    ah, al
3650                                     <1>      ; al = LVI, ah = CVI
3651                                     <1>      ;cmp     al, ah
3652                                     <1>      ;jne     short _ac97_ih5
3653                                     <1>      ; 03/06/2024
3654 000144B0 88C4                  <1>      mov     ah, al
3655                                     <1>
3656 000144B2 FEC8                  <1>      dec     al
3657 000144B4 241F                  <1>      and     al, 1Fh
3658                                     <1>
3659                                     <1>      ;mov     dx, PO_LVI_REG
3660                                     <1>      ;add     dx, [NABMBAR]
3661 000144B6 6642                  <1>      inc     dx ; 02/06/2024
3662 000144B8 EE                    <1>      out     dx, al
3663                                     <1>
3664                                     <1>      ; 12/10/2017
3665                                     <1>      ;mov     al, [audio_civ]
3666                                     <1>      ;inc     al
3667                                     <1>      ;and     al, 1
3668                                     <1>      ;mov     [audio_flag], al
3669                                     <1>      ; 27/11/2023
3670                                     <1>      ; 20/11/2023
3671                                     <1>      ;inc     ah
3672                                     <1>      _ac97_ih5: ; 02/06/2024
3673 000144B9 80E401              <1>      and     ah, 1
3674 000144BC 8825[FC880100]        <1>      mov     [audio_flag], ah
3675                                     <1>
3676                                     <1>      ;; [audio_flag] : 0 = Buffer 1, 1 = Buffer 2
3677                                     <1>      ;
3678                                     <1>      ; 21/11/2023
3679                                     <1>      ;call    ac97_tuneloop
3680                                     <1>
3681                                     <1>      ; 24/11/2023
3682                                     <1>      ac97_tuneloop:
3683                                     <1>      ; 09/10/2017
3684 000144C2 8B3D[F0880100]        <1>      mov     edi, [audio_dma_buff]
3685                                     <1>      ; 04/06/2024
3686                                     <1>      ;mov     ecx, [audio_dmabuff_size]
3687                                     <1>      ;shr     ecx, 1 ; dma buff size / 2 = half buffer size
3688 000144C8 8B0D[EC880100]        <1>      mov     ecx, [audio_buff_size]
3689 000144CE 80E1F8              <1>      and     cl, ~7 ; 8 byte aligned
3690                                     <1>
3691                                     <1>      ; 12/10/2017
3692 000144D1 803D[FC880100]00      <1>      cmp     byte [audio_flag], 0
3693 000144D8 7702                  <1>      ja      short _ac97_ih1 ; Playing Half Buffer 2 (Current: FLAG)
3694                                     <1>      ; Playing Half Buffer 1 (Current: EOL)
3695 000144DA 01CF                  <1>      add     edi, ecx
3696                                     <1>      _ac97_ih1:
3697                                     <1>      ; Update half buffer 2 while playing half buffer 1 (next: FLAG)
3698                                     <1>      ; Update half buffer 1 while playing half buffer 2 (next: EOL)
3699                                     <1>
3700 000144DC 8B35[E8880100]        <1>      mov     esi, [audio_p_buffer] ; phy addr of audio buff
3701 000144E2 C1E902              <1>      shr     ecx, 2 ; half buff size / 4
3702 000144E5 F3A5                  <1>      rep     movsd
3703                                     <1>
3704                                     <1>      ; 10/10/2017
3705                                     <1>      ; switch flag value
3706 000144E7 8035[FC880100]01      <1>      xor     byte [audio_flag], 1
3707                                     <1>
3708                                     <1>      ; 02/06/2024
3709                                     <1>      %if 1
3710                                     <1>      ; 02/06/2024
3711 000144EE 58                    <1>      pop     eax ; *
3712                                     <1>      _ac97_ih2:
3713                                     <1>      ;mov     ax, 1Ch ; FIFOE(=16)+BCIS(=8)+LVBCI(=4)
3714 000144EF 66BA1600              <1>      mov     dx, PO_SR_REG

```

```

3715 000144F3 660315[D6880100] <1> add dx, [NABMBAR]
3716 000144FA 66EF <1> out dx, ax
3717 <1> %else
3718 <1> ; 27/11/2023
3719 <1> ; 21/11/2023 - temporary
3720 <1> ;push ebx
3721 <1> ;mov ebx, 0B8002h
3722 <1> ;mov al, [audio_flag]
3723 <1> ;add al, '1'
3724 <1> ;mov ah, 0Fh
3725 <1> ;mov [ebx], ax
3726 <1> ;pop ebx
3727 <1>
3728 <1> ; 12/10/2017
3729 <1> ; [audio_flag] = 0 : Playing dma half buffer 2 (even index value)
3730 <1> ; ; Next buffer (to update) is dma half buff 1
3731 <1> ; = 1 : Playing dma half buffer 1 (odd index value)
3732 <1> ; ; Next buffer (to update) is dma half buff 2
3733 <1> ; 24/11/2023
3734 <1> ;retn
3735 <1>
3736 <1> ;
3737 <1> ; 24/11/2023
3738 <1> pop eax
3739 <1> ; 24/11/2023
3740 <1> ; 20/11/2023
3741 <1> ;and eax, 40h
3742 <1> mov dx, [NABMBAR]
3743 <1> add dx, GLOB_STS_REG
3744 <1> out dx, eax
3745 <1>
3746 <1> ;; 13/06/2017
3747 <1> ;mov al, 11h ; IOCE + RPBM
3748 <1> ;mov dx, PO_CR_REG
3749 <1> ;add dx, [NABMBAR]
3750 <1> ;out dx, al
3751 <1>
3752 <1> ; 24/11/2023
3753 <1> ;mov dx, [NABMBAR]
3754 <1> ;add dx, PO_SR_REG ; set pointer to Status reg
3755 <1> ;mov ax, 1Ch
3756 <1> ;out dx, ax
3757 <1>
3758 <1> ; 24/11/2023
3759 <1> _ac97_ih2:
3760 <1> ;mov byte [audio_busy], 0
3761 <1> %endif
3762 <1>
3763 <1> _ac97_ih3:
3764 <1> ;pop edi
3765 <1> ;pop esi
3766 <1> ;pop ebx ; * must be restored !
3767 <1> ;pop ecx
3768 <1> ;pop edx
3769 <1> ;pop eax ; * must be restored !
3770 <1>
3771 000144FC C3 <1> retn
3772 <1>
3773 <1> reset_ac97_controller:
3774 <1> ; 06/06/2024
3775 <1> ; 01/06/2024
3776 <1> ; 27/05/2024
3777 <1> ; 16/05/2024
3778 <1> ; 10/06/2017
3779 <1> ; 29/05/2017
3780 <1> ; 28/05/2017
3781 <1> ; reset AC97 audio controller registers
3782 000144FD 31C0 <1> xor eax, eax
3783 000144FF 66BA0B00 <1> mov dx, PI_CR_REG
3784 00014503 660315[D6880100] <1> add dx, [NABMBAR]
3785 0001450A EE <1> out dx, al
3786 <1>
3787 <1> ; 16/05/2024
3788 <1> ;call delay1_4ms
3789 <1>
3790 0001450B 66BA1B00 <1> mov dx, PO_CR_REG
3791 0001450F 660315[D6880100] <1> add dx, [NABMBAR]
3792 00014516 EE <1> out dx, al
3793 <1>
3794 <1> ; 16/05/2024
3795 <1> ;call delay1_4ms
3796 <1>
3797 00014517 66BA2B00 <1> mov dx, MC_CR_REG
3798 0001451B 660315[D6880100] <1> add dx, [NABMBAR]
3799 00014522 EE <1> out dx, al
3800 <1>
3801 <1> ; 16/05/2024
3802 <1> ;call delay1_4ms
3803 <1>
3804 00014523 B002 <1> mov al, RR
3805 00014525 66BA0B00 <1> mov dx, PI_CR_REG
3806 00014529 660315[D6880100] <1> add dx, [NABMBAR]
3807 00014530 EE <1> out dx, al
3808 <1>
3809 <1> ; 16/05/2024
3810 <1> ;call delay1_4ms
3811 <1>
3812 00014531 66BA1B00 <1> mov dx, PO_CR_REG
3813 00014535 660315[D6880100] <1> add dx, [NABMBAR]
3814 0001453C EE <1> out dx, al
3815 <1>
3816 <1> ; 16/05/2024
3817 <1> ;call delay1_4ms
3818 <1>
3819 0001453D 66BA2B00 <1> mov dx, MC_CR_REG
3820 00014541 660315[D6880100] <1> add dx, [NABMBAR]
3821 00014548 EE <1> out dx, al
3822 <1>
3823 <1> ; 27/05/2024
3824 <1> ; 16/05/2024
3825 <1> ;call delay1_4ms
3826 <1>
3827 00014549 C3 <1> retn
3828 <1>
3829 <1> ac97_reset:
3830 <1> ; 27/05/2024 - TRDOS 386 v2.0.8
3831 <1> ; 06/08/2022 - TRDOS 386 v2.0.5
3832 <1> ; 10/06/2017
3833 <1> ; 29/05/2017
3834 <1> ; 28/05/2017
3835 0001454A E8AEFFFFFF <1> call reset_ac97_controller
3836 <1> ; 29/05/2017
3837 <1> ;jmp reset_ac97_codec
3838 <1> ; 27/05/2024

```

```

3839 0001454F E80DF6FFFF <1> call delay_100ms
3840 <1> reset_ac97_codec:
3841 <1> ; 28/05/2017 - Erdogan Tan (Ref: kolibriOS, intelac97.asm)
3842 00014554 66BA2C00 <1> mov dx, GLOB_CNT_REG ; 2Ch
3843 00014558 660315[D6880100] <1> add dx, [NABMBAR]
3844 0001455F ED <1> in eax, dx
3845 <1>
3846 <1> ;test eax, 2
3847 <1> ; 06/08/2022
3848 00014560 A802 <1> test al, 2
3849 00014562 7407 <1> jz short _r_ac97codec_cold
3850 <1>
3851 00014564 E80F000000 <1> call warm_ac97codec_reset
3852 00014569 7308 <1> jnc short _r_ac97codec_ok
3853 <1> _r_ac97codec_cold:
3854 0001456B E83B000000 <1> call cold_ac97codec_reset
3855 00014570 7301 <1> jnc short _r_ac97codec_ok
3856 <1>
3857 <1> ; 16/04/2017
3858 <1> ;xor eax, eax ; timeout error
3859 <1> ;stc
3860 00014572 C3 <1> retn
3861 <1>
3862 <1> _r_ac97codec_ok:
3863 00014573 31C0 <1> xor eax, eax
3864 <1> ;mov al, VIA_ACLINK_C00_READY ; 1
3865 00014575 FEC0 <1> inc al
3866 00014577 C3 <1> retn
3867 <1>
3868 <1> warm_ac97codec_reset:
3869 <1> ; 06/08/2022 - TRDOS 386 v2.0.5
3870 <1> ; 28/05/2017 - Erdogan Tan (Ref: kolibriOS, intelac97.asm)
3871 <1> ;mov eax, 6
3872 <1> ; 06/08/2022
3873 00014578 29C0 <1> sub eax, eax
3874 0001457A B006 <1> mov al, 6
3875 0001457C 66BA2C00 <1> mov dx, GLOB_CNT_REG ; 2Ch
3876 00014580 660315[D6880100] <1> add dx, [NABMBAR]
3877 00014587 EF <1> out dx, eax
3878 <1>
3879 <1> ;mov ecx, 10; total 1s
3880 <1> ; 06/08/2022
3881 00014588 31C9 <1> xor ecx, ecx
3882 0001458A B10A <1> mov cl, 10
3883 <1> _warm_ac97c_rst_wait:
3884 0001458C 51 <1> push ecx
3885 0001458D E8CFF5FFFF <1> call delay_100ms
3886 00014592 59 <1> pop ecx
3887 <1>
3888 00014593 66BA3000 <1> mov dx, GLOB_STS_REG ; 30h
3889 00014597 660315[D6880100] <1> add dx, [NABMBAR]
3890 0001459E ED <1> in eax, dx
3891 <1>
3892 0001459F A900030010 <1> test eax, CTRL_ST_CREADY
3893 000145A4 7504 <1> jnz short _warm_ac97c_rst_ok
3894 <1>
3895 000145A6 49 <1> dec ecx
3896 000145A7 75E3 <1> jnz short _warm_ac97c_rst_wait
3897 <1>
3898 <1> _warm_ac97c_rst_fail:
3899 000145A9 F9 <1> stc
3900 <1> _warm_ac97c_rst_ok:
3901 000145AA C3 <1> retn
3902 <1>
3903 <1> cold_ac97codec_reset:
3904 <1> ; 06/08/2022 - TRDOS 386 v2.0.5
3905 <1> ; 28/05/2017 - Erdogan Tan (Ref: kolibriOS, intelac97.asm)
3906 <1> ;mov eax, 2
3907 <1> ; 06/08/2022
3908 000145AB 31C0 <1> xor eax, eax
3909 000145AD B002 <1> mov al, 2
3910 000145AF 66BA2C00 <1> mov dx, GLOB_CNT_REG ; 2Ch
3911 000145B3 660315[D6880100] <1> add dx, [NABMBAR]
3912 000145BA EF <1> out dx, eax
3913 <1>
3914 000145BB E8A1F5FFFF <1> call delay_100ms ; wait 100 ms
3915 000145C0 E89CF5FFFF <1> call delay_100ms ; wait 100 ms
3916 000145C5 E897F5FFFF <1> call delay_100ms ; wait 100 ms
3917 000145CA E892F5FFFF <1> call delay_100ms ; wait 100 ms
3918 <1>
3919 <1> ;mov ecx, 16; total 20*100 ms = 2s
3920 <1> ; 06/08/2022
3921 000145CF 31C9 <1> xor ecx, ecx
3922 000145D1 B110 <1> mov cl, 16
3923 <1> _cold_ac97c_rst_wait:
3924 000145D3 66BA3000 <1> mov dx, GLOB_STS_REG ; 30h
3925 000145D7 660315[D6880100] <1> add dx, [NABMBAR]
3926 000145DE ED <1> in eax, dx
3927 <1>
3928 000145DF A900030010 <1> test eax, CTRL_ST_CREADY
3929 000145E4 750B <1> jnz short _cold_ac97c_rst_ok
3930 <1>
3931 000145E6 51 <1> push ecx
3932 000145E7 E875F5FFFF <1> call delay_100ms
3933 000145EC 59 <1> pop ecx
3934 <1>
3935 000145ED 49 <1> dec ecx
3936 000145EE 75E3 <1> jnz short _cold_ac97c_rst_wait
3937 <1>
3938 <1> _cold_ac97c_rst_fail:
3939 000145F0 F9 <1> stc
3940 <1> _cold_ac97c_rst_ok:
3941 000145F1 C3 <1> retn
3942 <1>
3943 <1> sb16_current_sound_data:
3944 <1> ; 05/06/2024
3945 <1> ; 04/06/2024 - TRDOS 386 v2.0.8
3946 <1> ; 06/08/2022 - TRDOS 386 v2.0.5
3947 <1> ; 20/08/2017
3948 <1> ; 24/06/2017
3949 <1> ; 22/06/2017
3950 <1> ; get current sound (PCM out) data for graphics
3951 <1> ; (for Sound Blaster 16)
3952 <1> ; ebx = Physical address (on page boundary)
3953 <1> ; ecx = Byte count
3954 <1> ; [audio_buff_size]
3955 <1>
3956 <1> ;;mov edi, [audio_buff_size]
3957 <1> ;mov edi, [audio_dmabuff_size]
3958 <1> ;mov esi, [audio_dma_buff]
3959 <1>
3960 000145F2 39CF <1> cmp edi, ecx
3961 000145F4 7302 <1> jnb short sb16_gcd_0
3962 000145F6 89F9 <1> mov ecx, edi

```

```

3963 <1> sb16_gcd_0:
3964 <1> ; 06/08/2022
3965 000145F8 31C0 <1> xor     eax, eax
3966 <1> ; 20/08/2017
3967 000145FA 803D[04890100]10 <1> cmp     byte [audio_bps], 16
3968 00014601 750C <1> jne     short sb16_gcd_1 ; 8 bit DMA channel
3969 00014603 E4C6 <1> in      al, 0C6h ; DMA channel 5 count register
3970 <1> ;mov    dl, al
3971 <1> ; 06/08/2022
3972 00014605 88C4 <1> mov     ah, al
3973 00014607 E4C6 <1> in      al, 0C6h
3974 <1> ;mov    dh, al
3975 <1> ;movzx  eax, dx
3976 <1> ; 06/08/2022
3977 00014609 86C4 <1> xchg    ah, al
3978 0001460B D1E0 <1> shl     eax, 1 ; word count -> byte count
3979 0001460D EB4C <1> jmp     short sb16_gcd_2
3980 <1> sb16_gcd_1:
3981 0001460F E403 <1> in      al, 03h ; DMA channel 1 count register
3982 <1> ;mov    dl, al
3983 <1> ; 06/08/2022
3984 00014611 88C4 <1> mov     ah, al
3985 00014613 E403 <1> in      al, 03h
3986 <1> ;mov    dh, al
3987 <1> ;movzx  eax, dx
3988 <1> ; 06/08/2022
3989 00014615 86C4 <1> xchg    ah, al
3990 00014617 EB42 <1> jmp     short sb16_gcd_2
3991 <1> ;sb16_gcd_2:
3992 <1> ; cmp    eax, ecx
3993 <1> ; jnb    short sb16_gcd_3
3994 <1> ; ; remain count < graphics bytes
3995 <1> ; mov    eax, ecx ; fix remain count to data size
3996 <1> ;sb16_gcd_3:
3997 <1> ; sub    edi, eax
3998 <1> ; jna    short sb16_gcd_4
3999 <1> ; add    esi, edi ; dma buffer offset
4000 <1> ;sb16_gcd_4:
4001 <1> ; mov    edi, ebx ; buffer address (for graphics)
4002 <1> ; mov    [u.r0], ecx
4003 <1> ; rep    movsb
4004 <1> ; retn
4005 <1>
4006 <1> get_current_sound_data:
4007 <1> ; 05/06/2024
4008 <1> ; 04/06/2024
4009 <1> ; 24/06/2017
4010 <1> ; 22/06/2017
4011 <1> ; get current sound (PCM out) data for graphics
4012 <1> ;
4013 <1> ; ebx = Physical address (on page boundary)
4014 <1> ; ecx = Byte count
4015 <1> ; [audio_buff_size]
4016 <1>
4017 <1> ;mov     edi, [audio_buff_size]
4018 <1> ;;;
4019 <1> ; 04/06/2024
4020 <1> ;mov     edi, [audio_dmabuff_size]
4021 <1> ; 05/06/2024
4022 <1> ;mov     edi, [audio_buff_size]
4023 00014619 8B3D[F8880100] <1> mov     edi, [dma_hbuff_size]
4024 <1> ;;;
4025 0001461F 8B35[F0880100] <1> mov     esi, [audio_dma_buff]
4026 <1>
4027 <1> ;;;
4028 <1> ; 04/06/2024
4029 <1> ;cmp     byte [audio_device], 2
4030 <1> ;jb      short sb16_current_sound_data ; = 1
4031 <1> ;shr     edi, 1
4032 <1>
4033 <1> ; 04/06/2024
4034 00014625 803D[D1880100]01 <1> cmp     byte [audio_device], 1
4035 0001462C 7704 <1> ja      short gcd_1
4036 0001462E D1E7 <1> shl     edi, 1 ; [audio_dmabuff_size] = 2 * edi
4037 <1> ; 05/06/2024
4038 <1> ; edi = [dma_hbuff_size] * 2
4039 00014630 EBC0 <1> jmp     short sb16_current_sound_data
4040 <1> gcd_1:
4041 <1> ;;;
4042 <1>
4043 <1> ; 04/06/2024
4044 <1> %if 0
4045 <1> ; cmp    edi, ecx
4046 <1> ; jnb    short gcd_0
4047 <1> ; mov    ecx, edi
4048 <1> gcd_0:
4049 <1> %endif
4050 00014632 803D[D1880100]03 <1> cmp     byte [audio_device], 3
4051 00014639 7237 <1> jb      short ac97_current_sound_data ; = 2
4052 <1> ; = 3
4053 <1> vt8233_current_sound_data:
4054 <1> ; 05/06/2024
4055 <1> ; 04/06/2024 - TRDOS 386 v2.0.8
4056 <1> ; 06/08/2022 - TRDOS 386 v2.0.5
4057 <1> ; 22/06/2017
4058 <1> ; 21/06/2017
4059 <1> ; get current sound (PCM out) data for graphics
4060 <1> ; (for VT 8233, VT 8237R)
4061 <1> ; ebx = Physical address (on page boundary)
4062 <1> ; ecx = Byte count
4063 <1> ; [audio_buff_size]
4064 <1>
4065 <1> ;mov     edi, [audio_buff_size]
4066 <1> ;mov     edi, [audio_dmabuff_size]
4067 <1> ;mov     esi, [audio_dma_buff]
4068 <1> ;shr     edi, 1
4069 <1> ;cmp     edi, ecx
4070 <1> ;jnb     short vt8233_gcd_1
4071 <1> ;mov     ecx, edi
4072 <1>
4073 <1> ;;;
4074 <1> ; 04/06/2024
4075 <1> ; edi = [audio_buff_size]
4076 <1> ; 05/06/2024
4077 <1> ; and    di, ~1 ; word alignment
4078 <1> ; edi = [dma_hbuff_size]
4079 0001463B 39CF <1> cmp     edi, ecx
4080 0001463D 7302 <1> jnb     short vt8233_gcd_1
4081 0001463F 89F9 <1> mov     ecx, edi
4082 <1> ;;;
4083 <1>
4084 <1> vt8233_gcd_1:
4085 <1> ;mov     edx, VIA_REG_OFFSET_CURR_COUNT
4086 <1> ; 06/08/2022

```

```

4087 00014641 31D2      <1>      xor     edx, edx
4088 00014643 B20C      <1>      mov     dl, VIA_REG_OFFSET_CURR_COUNT
4089 00014645 E877F5FFFF <1>      call    ctrl_io_r32
4090 0001464A 89C2      <1>      mov     edx, eax ; remain count (bits 23-0),
4091                                ; SGD index (bits 31-24)
4092 0001464C 81E200000001 <1>      and     edx, 1000000h ; SGD index (0 = 1st half)
4093 00014652 7402      <1>      jz      short vt8233_gcd_2
4094                                ; the second half of DMA buffer
4095 00014654 01FE      <1>      add     esi, edi
4096                                vt8233_gcd_2:
4097 00014656 25FFFFFFF00 <1>      and     eax, 0FFFFFFh ; bits 23-0
4098                                ac97_gcd_2:
4099                                sb16_gcd_2:
4100 0001465B 39C8      <1>      cmp     eax, ecx
4101 0001465D 7302      <1>      jnb     short vt8233_gcd_3
4102                                ; remain count < graphics bytes
4103 0001465F 89C8      <1>      mov     eax, ecx ; fix remain count to data size
4104                                vt8233_gcd_3:
4105 00014661 29C7      <1>      sub     edi, eax
4106 00014663 7602      <1>      jna     short vt8233_gcd_4
4107 00014665 01FE      <1>      add     esi, edi ; dma buffer offset
4108                                vt8233_gcd_4:
4109 00014667 89DF      <1>      mov     edi, ebx ; buffer address (for graphics)
4110 00014669 890D[548E0100] <1>      mov     [u.r0], ecx
4111 0001466F F3A4      <1>      rep     movsb
4112                                vt8233_gcd_5:
4113 00014671 C3      <1>      retn
4114                                ;
4115                                ac97_current_sound_data:
4116                                ; 05/06/2024
4117                                ; 04/06/2024 - TRDOS 386 v2.0.8
4118                                ; 23/06/2017
4119                                ; 22/06/2017
4120                                ; get current sound (PCM out) data for graphics
4121                                ; (for AC'97, ICH)
4122                                ; ebx = Physical address (on page boundary)
4123                                ; ecx = Byte count
4124                                ; [audio_buff_size]
4125                                ;
4126                                ; mov edi, [audio_buff_size]
4127                                ; mov edi, [audio_dmabuff_size]
4128                                ; mov esi, [audio_dma_buff]
4129                                ; shr edi, 1
4130                                ; cmp edi, ecx
4131                                ; jnb short ac97_gcd_0
4132                                ; mov ecx, edi
4133                                ;
4134                                ;
4135                                ; 04/06/2024
4136                                ; edi = [audio_buff_size]
4137                                ; 05/06/2024
4138                                ; and di, ~7 ; 8 byte alignment
4139                                ; edi = [dma_hbuff_size]
4140 00014672 39CF      <1>      cmp     edi, ecx
4141 00014674 7302      <1>      jnb     short ac97_gcd_0
4142 00014676 89F9      <1>      mov     ecx, edi
4143                                ;
4144                                ;
4145                                ac97_gcd_0:
4146 00014678 66BA1400 <1>      mov     dx, PO_CIV_REG ; Position In Current Buff Reg
4147 0001467C 660315[D6880100] <1>      add     dx, [NABMBAR]
4148 00014683 EC      <1>      in      al, dx ; current index value
4149 00014684 A801      <1>      test    al, 1
4150 00014686 7402      <1>      jz      short ac97_gcd_1
4151 00014688 01FE      <1>      add     esi, edi
4152                                ac97_gcd_1:
4153 0001468A 31C0      <1>      xor     eax, eax
4154 0001468C 66BA1800 <1>      mov     dx, PO_PICB_REG ; Position In Current Buff Reg
4155 00014690 660315[D6880100] <1>      add     dx, [NABMBAR]
4156                                ; in ax, dx ; remain dwords
4157                                ; shl eax, 2 ; remain bytes ; 23/06/2017
4158                                ;
4159                                ; 04/06/2024
4160 00014697 66ED      <1>      in      ax, dx ; remain words
4161 00014699 D1E0      <1>      shl     eax, 1 ; remain bytes
4162                                ;
4163 0001469B EBBE      <1>      jmp     short ac97_gcd_2
4164                                ; cmp eax, ecx
4165                                ; jnb short ac97_gcd_2
4166                                ; ; remain count < graphics bytes
4167                                ; mov eax, ecx ; fix remain count to data size
4168                                ; ac97_gcd_2:
4169                                ; sub edi, eax
4170                                ; jna short ac97_gcd_3
4171                                ; add esi, edi ; dma buffer offset
4172                                ; ac97_gcd_3:
4173                                ; mov edi, ebx ; buffer address (for graphics)
4174                                ; mov [u.r0], ecx
4175                                ; rep movsb
4176                                ; retn
4177                                ;
4178                                sb16_get_dma_buff_off:
4179                                ; 05/06/2024
4180                                ; 04/06/2024 - TRDOS 386 v2.0.8
4181                                ; 28/10/2017
4182                                ; 24/06/2017
4183                                ; 22/06/2017
4184                                ; get current (PCM OUT DMA buffer) pointer
4185                                ; (for Sound Blaster 16)
4186                                ;
4187                                ; mov ecx, [audio_dmabuff_size]
4188                                ; xor ebx, ebx
4189                                ; shr ecx, 1
4190                                ;
4191                                ;
4192                                ; 04/06/2024
4193                                ; ecx = audio buffer size
4194                                ; 05/06/2024
4195                                ; ecx = DMA half buffer size
4196 0001469D D1E1      <1>      shl     ecx, 1 ; * 2
4197                                ; ecx = DMA buffer size
4198                                ;
4199                                ;
4200                                sb16_gdmabo_0:
4201                                ; 28/10/2017
4202 0001469F 803D[04890100]10 <1>      cmp     byte [audio_bps], 16
4203 000146A6 750F      <1>      jne     short sb16_gdmabo_1 ; 8 bit DMA channel
4204                                ; 16 bit DMA channel
4205 000146A8 E4C6      <1>      in      al, 0C6h ; DMA channel 5 count register
4206 000146AA 88C2      <1>      mov     dl, al
4207 000146AC E4C6      <1>      in      al, 0C6h
4208 000146AE 88C6      <1>      mov     dh, al
4209 000146B0 0FB7C2 <1>      movzx   eax, dx
4210 000146B3 D1E0      <1>      shl     eax, 1 ; word count -> byte count

```

```

4211 000146B5 EB3A      <1>      jmp      short sb16_gdmabo_2
4212                  <1> sb16_gdmabo_1:
4213 000146B7 E403      <1>      in       al, 03h ; DMA channel 1 count register
4214 000146B9 88C2      <1>      mov     dl, al
4215 000146BB E403      <1>      in       al, 03h
4216 000146BD 88C6      <1>      mov     dh, al
4217 000146BF 0FB7C2     <1>      movzx   eax, dx
4218 000146C2 EB2D      <1>      jmp     short sb16_gdmabo_2
4219                  <1>
4220                  <1> get_dma_buffer_offset:
4221                  <1>      ; 05/06/2024
4222                  <1>      ; 04/06/2024
4223                  <1>      ; 24/06/2017
4224                  <1>      ; 22/06/2017
4225                  <1>      ; get current sound (PCM out) data for graphics
4226                  <1>      ;
4227                  <1>      ; ebx = Physical address (on page boundary)
4228                  <1>      ; ecx = Byte count
4229                  <1>      ; [audio_buff_size]
4230                  <1>
4231                  <1>      ;mov    ecx, [audio_dmabuff_size]
4232                  <1>      ;;;
4233                  <1>      ; 04/06/2024
4234                  <1>      ;mov    ecx, [audio_buff_size]
4235                  <1>      ; 05/06/2024
4236 000146C4 8B0D[F880100] <1>      mov     ecx, [dma_hbuff_size]
4237                  <1>      ;;;
4238 000146CA 31DB      <1>      xor     ebx, ebx
4239                  <1> gdmabo_0:
4240 000146CC 803D[D1880100]02 <1>      cmp     byte [audio_device], 2
4241 000146D3 72C8      <1>      jb      short sb16_get_dma_buff_off
4242 000146D5 7427      <1>      je      short ac97_get_dma_buff_off
4243                  <1>
4244                  <1> vt8233_get_dma_buff_off:
4245                  <1>      ; 05/06/2024
4246                  <1>      ; 04/06/2024 - TRDOS 386 v2.0.8
4247                  <1>      ; 06/08/2022 - TRDOS 386 v2.0.5
4248                  <1>      ; 24/06/2017
4249                  <1>      ; 22/06/2017
4250                  <1>      ; get current (PCM OUT DMA buffer) pointer
4251                  <1>      ; (for VT 8233, VT 8237R)
4252                  <1>
4253                  <1>      ;mov    ecx, [audio_dmabuff_size]
4254                  <1>      ;xor    ebx, ebx
4255                  <1>      ;;;
4256                  <1>      ; 04/06/2024
4257                  <1>      ;shr    ecx, 1
4258                  <1>      ; 05/06/2024
4259                  <1>      ;and    cl, ~1
4260                  <1>      ; ecx = DMA Half Buffer Size (word aligned)
4261                  <1>      ;;;
4262                  <1> vt8233_gdmabo_0:
4263                  <1>      ;mov    edx, VIA_REG_OFFSET_CURR_COUNT
4264                  <1>      ; 06/08/2022
4265 000146D7 31D2      <1>      xor     edx, edx
4266 000146D9 B20C      <1>      mov     dl, VIA_REG_OFFSET_CURR_COUNT
4267 000146DB E8E1F4FFFF <1>      call    ctrl_io_r32
4268 000146E0 89C2      <1>      mov     edx, eax ; remain count (bits 23-0),
4269                  <1>      ; SGD index (bits 31-24)
4270 000146E2 81E200000001 <1>      and     edx, 1000000h ; SGD index (0 = 1st half)
4271 000146E8 7402      <1>      jz      short vt8233_gdmabo_1
4272                  <1>      ; the second half of DMA buffer
4273 000146EA 89CB      <1>      mov     ebx, ecx
4274                  <1> vt8233_gdmabo_1:
4275 000146EC 25FFFFFFF00 <1>      and     eax, 0FFFFFFh ; bits 23-0
4276                  <1> sb16_gdmabo_2:
4277                  <1> ac97_gdmabo_2:
4278 000146F1 29C1      <1>      sub     ecx, eax
4279 000146F3 7602      <1>      jna     short vt8233_gdmabo_2
4280 000146F5 01CB      <1>      add     ebx, ecx ; dma buffer offset
4281                  <1> vt8233_gdmabo_2:
4282 000146F7 891D[548E0100] <1>      mov     [u.r0], ebx
4283 000146FD C3        <1>      retn
4284                  <1>
4285                  <1> ac97_get_dma_buff_off:
4286                  <1>      ; 06/06/2024
4287                  <1>      ; 05/06/2024
4288                  <1>      ; 04/06/2024 - TRDOS 386 v2.0.8
4289                  <1>      ; 24/06/2017
4290                  <1>      ; 22/06/2017
4291                  <1>      ; get current (PCM OUT DMA buffer) pointer
4292                  <1>      ; (for AC'97, ICH)
4293                  <1>      ; ebx = Physical address (on page boundary)
4294                  <1>      ; ecx = Byte count
4295                  <1>      ; [audio_buff_size]
4296                  <1>
4297                  <1>      ;mov    ecx, [audio_dmabuff_size]
4298                  <1>      ;xor    ebx, ebx
4299                  <1>      ;;;
4300                  <1>      ; 04/06/2024
4301                  <1>      ;shr    ecx, 1
4302                  <1>      ; 05/06/2024
4303                  <1>      ;and    cl, ~7 ; (truncate bytes if out of 8x)
4304                  <1>      ; ecx = DMA Half Buffer Size (8 byte aligned)
4305                  <1>      ;;;
4306                  <1> ac97_gdmabo_0:
4307 000146FE 66BA1400 <1>      mov     dx, PO_CIV_REG ; Position In Current Buff Reg
4308 00014702 660315[D6880100] <1>      add     dx, [NABMBAR]
4309 00014709 EC        <1>      in       al, dx ; current index value
4310 0001470A A801      <1>      test    al, 1
4311 0001470C 7402      <1>      jz      short ac97_gdmabo_1
4312 0001470E 89CB      <1>      mov     ebx, ecx
4313                  <1> ac97_gdmabo_1:
4314 00014710 31C0      <1>      xor     eax, eax
4315 00014712 66BA1800 <1>      mov     dx, PO_PICB_REG ; Position In Current Buff Reg
4316 00014716 660315[D6880100] <1>      add     dx, [NABMBAR]
4317 0001471D 66ED      <1>      in       ax, dx ; remain words (samples)
4318                  <1>      ;;;
4319                  <1>      ; 06/06/2024
4320                  <1>      ; audio samples are counted as words
4321                  <1>      ; 04/06/2024 (BugFix)
4322 0001471F D1E0      <1>      shl     eax, 1 ; remain bytes
4323                  <1>      ;;;
4324 00014721 EBCE      <1>      jmp     short ac97_gdmabo_2
4325                  <1>
4326                  <1> ac97_codec_info:
4327                  <1>      ; 06/06/2024
4328                  <1>      ; 05/06/2024 - TRDOS 386 v2.0.8
4329                  <1>      ; 19/11/2023
4330                  <1>      ; ENTRY: none
4331                  <1>      ; RETURN:
4332                  <1>      ;     eax = Extended Audio ID (MX28) in ax
4333                  <1>      ;     ax bit 0 - VRA bit
4334                  <1>      ;     hw of eax = PCM output sample rate (48000)

```

```

4335 <1> ; ebx = VENDOR ID 1, VENDOR ID 2 (bx)
4336 <1> ; cf = 1 -> error
4337 <1>
4338 <1> ; 06/06/2024
4339 <1> ;;;
4340 00014723 66BA3000 <1> mov dx, GLOB_STS_REG ; 30h
4341 00014727 660315[D6880100] <1> add dx, [NABMBAR]
4342 0001472E ED <1> in eax, dx
4343 <1>
4344 <1> ;cmp eax, 0FFFFFFFh ; -1
4345 0001472F 40 <1> inc eax
4346 00014730 7408 <1> jz short ac97_c_inf_err
4347 00014732 48 <1> dec eax
4348 <1>
4349 00014733 2500030010 <1> and eax, CTRL_ST_CREADY
4350 00014738 7502 <1> jnz short ac97_c_inf_1
4351 <1>
4352 <1> ac97_c_inf_err:
4353 0001473A F9 <1> stc
4354 0001473B C3 <1> retn
4355 <1> ;;;
4356 <1>
4357 <1> ac97_c_inf_1:
4358 0001473C 668B15[D4880100] <1> mov dx, [NABMBAR]
4359 00014743 6683C22C <1> add dx, CODEC_PCM_FRONT_DACRATE_REG
4360 00014747 66ED <1> in ax, dx ; PCM Output Sample Rate
4361 <1> ; 48000 Hz
4362 00014749 C1E010 <1> shl eax, 16
4363 0001474C E810F4FFFF <1> call delay_100ms
4364 00014751 668B15[D4880100] <1> mov dx, [NABMBAR]
4365 00014758 6683C228 <1> add dx, CODEC_EXT_AUDIO_REG ; 28h
4366 0001475C 66ED <1> in ax, dx
4367 0001475E 50 <1> push eax
4368 0001475F E8FDF3FFFF <1> call delay_100ms
4369 00014764 668B15[D4880100] <1> mov dx, [NABMBAR]
4370 0001476B 6683C27C <1> add dx, CODEC_VENDOR_ID1
4371 0001476F 66ED <1> in ax, dx
4372 00014771 C1E010 <1> shl eax, 16
4373 00014774 668B15[D4880100] <1> mov dx, [NABMBAR]
4374 0001477B 6683C27E <1> add dx, CODEC_VENDOR_ID2
4375 0001477F 66ED <1> in ax, dx
4376 00014781 89C3 <1> mov ebx, eax
4377 00014783 58 <1> pop eax
4378 <1> ;;;
4379 <1> ; 06/06/2024 (temporary!?)
4380 <1> ; (ALC850 & ALC655 BugFix)
4381 00014784 83FBFF <1> cmp ebx, 0FFFFFFFh ; -1
4382 00014787 74B1 <1> je short ac97_c_inf_err ; invalid
4383 00014789 09DB <1> or ebx, ebx
4384 0001478B 74AD <1> jz short ac97_c_inf_err ; invalid
4385 0001478D 81FB90474C41 <1> cmp ebx, 414C4790h ; ALC850
4386 <1> ;je short ac97_c_inf_2
4387 <1> ;cmp ebx, 414C4760h ; ALC655
4388 00014793 7505 <1> jne short ac97_c_inf_3
4389 <1> ;mov eax, 0BB8009C6h ; AC655 default (Read only)
4390 <1> ;retn
4391 <1> ac97_c_inf_2:
4392 <1> ; ref: ALC850 & ALC655 data sheet
4393 00014795 B8C40980BB <1> mov eax, 0BB8009C4h ; AC850 default (Read only)
4394 <1> ac97_c_inf_3:
4395 <1> ;;;
4396 0001479A C3 <1> retn
4397 <1>
4398 <1> ; 19/11/2023
4399 <1> ; valid ICH device IDs
4400 <1>
4401 <1> valid_ids:
4402 0001479B 86801524 <1> dd (ICH_DID << 16) + INTEL_VID ; 8086h:2415h
4403 0001479F 86802524 <1> dd (ICH0_DID << 16) + INTEL_VID ; 8086h:2425h
4404 000147A3 86804524 <1> dd (ICH2_DID << 16) + INTEL_VID ; 8086h:2445h
4405 000147A7 86808524 <1> dd (ICH3_DID << 16) + INTEL_VID ; 8086h:2485h
4406 000147AB 8680C524 <1> dd (ICH4_DID << 16) + INTEL_VID ; 8086h:24C5h
4407 000147AF 8680D524 <1> dd (ICH5_DID << 16) + INTEL_VID ; 8086h:24D5h
4408 000147B3 86806E26 <1> dd (ICH6_DID << 16) + INTEL_VID ; 8086h:266Eh
4409 000147B7 8680A625 <1> dd (ESB6300_DID << 16) + INTEL_VID ; 8086h:25A6h
4410 000147BB 86809826 <1> dd (ESB631X_DID << 16) + INTEL_VID ; 8086h:2698h
4411 000147BF 8680DE27 <1> dd (ICH7_DID << 16) + INTEL_VID ; 8086h:27DEh
4412 000147C3 86809571 <1> dd (MX82440_DID << 16) + INTEL_VID ; 8086h:7195h
4413 000147C7 39101270 <1> dd (SI7012_DID << 16) + SIS_VID ; 1039h:7012h
4414 000147CB DE10B101 <1> dd (NFORCE_DID << 16) + NVIDIA_VID ; 10DEh:01B1h
4415 000147CF DE106A00 <1> dd (NFORCE2_DID << 16) + NVIDIA_VID ; 10DEh:006Ah
4416 000147D3 22106D74 <1> dd (AMD8111_DID << 16) + AMD_VID ; 1022h:746Dh
4417 000147D7 22104574 <1> dd (AMD768_DID << 16) + AMD_VID ; 1022h:7445h
4418 000147DB DE105900 <1> dd (CK804_DID << 16) + NVIDIA_VID ; 10DEh:0059h
4419 000147DF DE103A00 <1> dd (MCP04_DID << 16) + NVIDIA_VID ; 10DEh:003Ah
4420 000147E3 DE108A00 <1> dd (CK8_DID << 16) + NVIDIA_VID ; 1022h:008Ah
4421 000147E7 DE10DA00 <1> dd (NFORCE3_DID << 16) + NVIDIA_VID ; 10DEh:00DAh
4422 000147EB DE10EA00 <1> dd (CK8S_DID << 16) + NVIDIA_VID ; 10DEh:00EAh
4423 <1>
4424 <1> valid_id_count: equ ($ - valid_ids)>>2 ; 19/11/2023
3787
3788 000147EF 90 align 4
3789
3790 %include 'vgadata.s' ; 04/07/2016
1 <1> ; *****
2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.3 - vgadata.s (palette and font data)
3 <1> ; -----
4 <1> ; Last Update: 01/01/2021
5 <1> ; -----
6 <1> ; Beginning: 16/01/2016
7 <1> ; -----
8 <1> ; Assembler: NASM version 2.15 (trdos386.s)
9 <1> ; -----
10 <1> ; Turkish Rational DOS
11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
12 <1> ;
13 <1> ; Derived from 'Plex86/Bochs VGABios' source code, vgabios-0.7a, (2011)
14 <1> ; by the LGPL VGABios Developers Team (2001-2008), 'vgatables.h'
15 <1> ;
16 <1> ; Oracle VirtualBox 5.0.24 VGABios Source Code
17 <1> ; ('vgabios.c', 'vgatables.h', 'vgafonts.h', 'vgarom.asm')
18 <1> ;
19 <1> ; Palette and font data in assembly language format:
20 <1> ; 'VBoxVgaBiosAlternative.asm'
21 <1> ;
22 <1> ; *****
23 <1> ;
24 <1> ; 25/11/2020 (TRDOS 386 v2.0.3)
25 <1> ; ('vgatables.h' - 30/12/2019 - vruppert)
26 <1> ;
27 <1> ; 04/07/2016
28 <1> ; COLOR DATA
29 <1> ;
30 <1> palette0: ; (63+1)*3

```



31	000147F0	000000000000000000-	<1>	db	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	
32	000147F9	000000000000000000	<1>																
33	00014800	000000000000000002A-	<1>	db	000h,	000h,	000h,	000h,	000h,	000h,	000h,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah	
34	00014809	2A2A2A2A2A2A2A2A	<1>																
35	00014810	2A2A2A2A2A2A2A2A2A-	<1>	db	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah	
36	00014819	2A2A2A2A2A2A2A2A	<1>																
37	00014820	2A2A2A2A2A2A2A2A2A-	<1>	db	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah	
38	00014829	2A2A2A2A2A2A2A2A	<1>																
39	00014830	2A2A2A2A2A2A2A2A3F-	<1>	db	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh	
40	00014839	3F3F3F3F3F3F3F3F	<1>																
41	00014840	3F3F3F3F3F3F3F3F3F-	<1>	db	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh	
42	00014849	3F3F3F3F3F3F3F3F	<1>																
43	00014850	0000000000000000000-	<1>	db	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h,	000h	
44	00014859	000000000000000000	<1>																
45	00014860	000000000000000002A-	<1>	db	000h,	000h,	000h,	000h,	000h,	000h,	000h,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah	
46	00014869	2A2A2A2A2A2A2A2A	<1>																
47	00014870	2A2A2A2A2A2A2A2A2A-	<1>	db	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah	
48	00014879	2A2A2A2A2A2A2A2A	<1>																
49	00014880	2A2A2A2A2A2A2A2A2A-	<1>	db	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah	
50	00014889	2A2A2A2A2A2A2A2A	<1>																
51	00014890	2A2A2A2A2A2A2A2A3F-	<1>	db	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	02ah,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh	
52	00014899	3F3F3F3F3F3F3F3F	<1>																
53	000148A0	3F3F3F3F3F3F3F3F3F-	<1>	db	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh,	03fh	
54	000148A9	3F3F3F3F3F3F3F3F	<1>																
55			<1>	palette1: ; (63+1)*3															
56	000148B0	000000000002A002A00-	<1>	db	000h,	000h,	000h,	000h,	000h,	02ah,	000h,	02ah,	000h,	000h,	02ah,	02ah,	000h,	000h,	02ah
57	000148B9	002A2A2A000002A	<1>																
58	000148C0	002A2A15002A2A2A00-	<1>	db	000h,	02ah,	02ah,	015h,	000h,	02ah,	02ah								

```
94 00014BB9 180E1C1C0E1C1C <1>
95 00014BC0 0E181C0E151C0E111C- <1> db 00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h
95 00014BC9 0E0E1C110E1C15 <1>
96 00014BD0 0E1C180E1C1C0E181C- <1> db 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh, 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 00eh
96 00014BD9 0E151C0E111C0E <1>
97 00014BE0 0E1C0E0E1C110E1C15- <1> db 00eh, 01ch, 00eh, 00eh, 01ch, 011h, 00eh, 01ch, 015h, 00eh, 01ch, 018h, 00eh, 01ch, 01ch, 00eh
97 00014BE9 0E1C180E1C1C0E <1>
98 00014BF0 181C0E151C0E111C14- <1> db 018h, 01ch, 00eh, 015h, 01ch, 00eh, 011h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h
98 00014BF9 141C16141C1814 <1>
99 00014C00 1C1A141C1C141C1C14- <1> db 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h
99 00014C09 1A1C14181C1416 <1>
100 00014C10 1C14141C16141C1814- <1> db 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch, 018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah
100 00014C19 1C1A141C1C141A <1>
101 00014C20 1C14181C14161C1414- <1> db 01ch, 014h, 018h, 01ch, 014h, 016h, 01ch, 014h, 014h, 01ch, 014h, 014h, 01ch, 016h, 014h, 01ch
101 00014C29 1C14141C16141C <1>
102 00014C30 18141C1A141C1C141A- <1> db 018h, 014h, 01ch, 01ah, 014h, 01ch, 01ch, 014h, 01ah, 01ch, 014h, 018h, 01ch, 014h, 016h, 01ch
102 00014C39 1C14181C14161C <1>
103 00014C40 000010040010080010- <1> db 000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h, 010h, 010h
103 00014C49 0C001010001010 <1>
104 00014C50 000C10000810000410- <1> db 000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h
104 00014C59 00001004001008 <1>
105 00014C60 00100C001010000C10- <1> db 000h, 010h, 00ch, 000h, 010h, 010h, 000h, 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 000h
105 00014C69 00081000041000 <1>
106 00014C70 001000001004001008- <1> db 000h, 010h, 000h, 000h, 010h, 004h, 000h, 010h, 008h, 000h, 010h, 00ch, 000h, 010h, 010h, 000h
106 00014C79 00100C00101000 <1>
107 00014C80 0C1000081000041008- <1> db 00ch, 010h, 000h, 008h, 010h, 000h, 004h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h
107 00014C89 08100A08100C08 <1>
108 00014C90 100E08101008101008- <1> db 010h, 00eh, 008h, 010h, 010h, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah
108 00014C99 0E10080C10080A <1>
109 00014CA0 100808100A08100C08- <1> db 010h, 008h, 008h, 010h, 00ah, 008h, 010h, 00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh
109 00014CA9 100E081010080E <1>
110 00014CB0 10080C10080A100808- <1> db 010h, 008h, 00ch, 010h, 008h, 00ah, 010h, 008h, 008h, 010h, 008h, 008h, 010h, 00ah, 008h, 010h
110 00014CB9 100808100A0810 <1>
111 00014CC0 0C08100E081010080E- <1> db 00ch, 008h, 010h, 00eh, 008h, 010h, 010h, 008h, 00eh, 010h, 008h, 00ch, 010h, 008h, 00ah, 010h
111 00014CC9 10080C10080A10 <1>
112 00014CD0 080B100C08100D0B10- <1> db 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 010h, 010h
112 00014CD9 0F0B10100B1010 <1>
113 00014CE0 080F100B0D100B0C10- <1> db 00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh
113 00014CE9 080B100C08100D <1>
114 00014CF0 08100F0B10100B0F10- <1> db 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh, 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 00bh
114 00014CF9 080D100B0C1008 <1>
115 00014D00 08100B08100C08100D- <1> db 00bh, 010h, 00bh, 00bh, 010h, 00ch, 00bh, 010h, 00dh, 00bh, 010h, 00fh, 00bh, 010h, 010h, 00bh
115 00014D09 08100F0B10100B <1>
116 00014D10 0F100B0D100B0C1000- <1> db 00fh, 010h, 00bh, 00dh, 010h, 00bh, 00ch, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
116 00014D19 0000000000000000 <1>
117 00014D20 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
117 00014D29 0000000000000000 <1>
118 <1>
119 <1>
120 <1> ; 04/07/2016
121 <1> ; FONT DATA
122 <1>
123 <1> CRT_CHAR_GEN:
124 <1> vgaFont8:
125 00014D30 000000000000000007E- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 081h, 0a5h, 081h, 0bdh, 099h, 081h, 07eh
125 00014D39 81A581BD99817E <1>
126 00014D40 7EFFDBFFC3E7FF7E6C- <1> db 07eh, 0ffh, 0dbh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 06ch, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h
126 00014D49 FEFEF7C381000 <1>
127 00014D50 10387CFE7C38100038- <1> db 010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 038h, 07ch, 038h, 0feh, 0feh, 07ch, 038h, 07ch
127 00014D59 7C38FEFE7C387C <1>
128 00014D60 1010387CFE7C387C00- <1> db 010h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 07ch, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h
128 00014D69 00183C3C180000 <1>
129 00014D70 FFFFE7C3C3E7FFFF00- <1> db 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h
129 00014D79 3C664242663C00 <1>
130 00014D80 FFC399BDBD99C3FF0F- <1> db 0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 00fh, 007h, 00fh, 07dh, 0cch, 0cch, 0cch, 078h
130 00014D89 070F7DCCCCC78 <1>
131 00014D90 3C6666663C187E183F- <1> db 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 03fh, 033h, 03fh, 030h, 030h, 070h, 0f0h, 0e0h
131 00014D99 333F303070F0E0 <1>
132 00014DA0 7F637F636367E6C099- <1> db 07fh, 063h, 07fh, 063h, 063h, 067h, 0e6h, 0c0h, 099h, 05ah, 03ch, 0e7h, 0e7h, 03ch, 05ah, 099h
132 00014DA9 5A3CE7E73C5A99 <1>
133 00014DB0 80E0F8FEF8E0800002- <1> db 080h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 080h, 000h, 002h, 00eh, 03eh, 0feh, 03eh, 00eh, 002h, 000h
133 00014DB9 0E3EFE3E0E0200 <1>
134 00014DC0 183C7E18187E3C1866- <1> db 018h, 03ch, 07eh, 018h, 018h, 07eh, 03ch, 018h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 000h
134 00014DC9 66666666006600 <1>
135 00014DD0 7FDBD87B1B1B1B003E- <1> db 07fh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 000h, 03eh, 063h, 038h, 06ch, 06ch, 038h, 0cch, 078h
135 00014DD9 63386C6C38CC78 <1>
136 00014DE0 000000007E7E7E0018- <1> db 000h, 000h, 000h, 000h, 07eh, 07eh, 07eh, 000h, 018h, 03ch, 07eh, 018h, 07eh, 03ch, 018h, 0ffh
136 00014DE9 3C7E187E3C18FF <1>
137 00014DF0 183C7E181818180018- <1> db 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h
137 00014DF9 1818187E3C1800 <1>
138 00014E00 00180CFE0C18000000- <1> db 000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h
138 00014E09 3060FE60300000 <1>
139 00014E10 0000C0C0C0FE000000- <1> db 000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h
139 00014E19 2466FF66240000 <1>
140 00014E20 00183C7EFFFF000000- <1> db 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 000h, 000h, 000h, 0ffh, 0ffh, 07eh, 03ch, 018h, 000h, 000h
140 00014E29 FFFF7E3C180000 <1>
141 00014E30 0000000000000000030- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 078h, 078h, 030h, 030h, 000h, 030h, 000h
141 00014E39 78783030003000 <1>
142 00014E40 6C6C6C000000000006C- <1> db 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 0feh, 06ch, 06ch, 000h
142 00014E49 6CFE6CFE6C6C00 <1>
143 00014E50 307CC0780CF8300000- <1> db 030h, 07ch, 0c0h, 078h, 00ch, 0f8h, 030h, 000h, 000h, 0c6h, 0cch, 018h, 030h, 066h, 0c6h, 000h
143 00014E59 6CCC183066C600 <1>
144 00014E60 386C3876DCCC760060- <1> db 038h, 06ch, 038h, 076h, 0dch, 0cch, 076h, 000h, 060h, 060h, 0c0h, 000h, 000h, 000h, 000h, 000h
144 00014E69 6C000000000000 <1>
145 00014E70 183060606030180060- <1> db 018h, 030h, 060h, 060h, 060h, 030h, 018h, 000h, 060h, 030h, 018h, 018h, 018h, 030h, 060h, 000h
145 00014E79 30181818306000 <1>
146 00014E80 00663CFF3C66000000- <1> db 000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 000h
146 00014E89 3030FC30300000 <1>
147 00014E90 000000000030306000- <1> db 000h, 000h, 000h, 000h, 000h, 030h, 030h, 060h, 000h, 000h, 000h, 0fch, 000h, 000h, 000h, 000h
147 00014E99 0000FC00000000 <1>
148 00014EA0 000000000030300006- <1> db 000h, 000h, 000h, 000h, 000h, 030h, 030h, 000h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h
148 00014EA9 0C183060C08000 <1>
149 00014EB0 7CC6CEDEF6E67C0030- <1> db 07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 07ch, 000h, 030h, 070h, 030h, 030h, 030h, 030h, 0fch, 000h
149 00014EB9 7030303030FC00 <1>
150 00014EC0 78CC0C3860CCFC0078- <1> db 078h, 0cch, 00ch, 038h, 060h, 0cch, 0fch, 000h, 078h, 0cch, 00ch, 038h, 00ch, 0cch, 078h, 000h
150 00014EC9 CC0C380CCC7800 <1>
151 00014ED0 1C3C6CCCFE0C1E00FC- <1> db 01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 01eh, 000h, 0fch, 0c0h, 0f8h, 00ch, 00ch, 0cch, 078h, 000h
151 00014ED9 C0F80C0CCC7800 <1>
152 00014EE0 3860C0F8CCCC7800FC- <1> db 038h, 060h, 0c0h, 0f8h, 0cch, 0cch, 078h, 000h, 0fch, 0cch, 00ch, 018h, 030h, 030h, 030h, 000h
152 00014EE9 CC0C1830303000 <1>
153 00014EF0 78CCCC78CCCC780078- <1> db 078h, 0cch, 0cch, 078h, 0cch, 0cch, 078h, 000h, 078h, 0cch, 0cch, 07ch, 00ch, 018h, 070h, 000h
153 00014EF9 CCCC7C0C187000 <1>
154 00014F00 003030000030300000- <1> db 000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 000h, 000h, 030h, 030h, 060h
154 00014F09 30300000303060 <1>
155 00014F10 183060C06030180000- <1> db 018h, 030h, 060h, 0c0h, 060h, 030h, 018h, 000h, 000h, 000h, 0fch, 000h, 000h, 0fch, 000h, 000h
155 00014F19 00FC0000FC0000 <1>
156 00014F20 6030180C1830600078- <1> db 060h, 030h, 018h, 00ch, 018h, 030h, 060h, 000h, 078h, 0cch, 00ch, 018h, 030h, 000h, 030h, 000h
156 00014F29 CC0C1830003000 <1>
157 00014F30 7CC6DEDEDEC0780030- <1> db 07ch, 0c6h, 0deh, 0deh, 0deh, 0c0h, 078h, 000h, 030h, 078h, 0cch, 0cch, 0fch, 0cch, 0cch, 000h
157 00014F39 78CCCCFCCCCC00 <1>
158 00014F40 FC66667C6666FC003C- <1> db 0fch, 066h, 066h, 07ch, 066h, 066h, 0fch, 000h, 03ch, 066h, 0c0h, 0c0h, 0c0h, 066h, 03ch, 000h
158 00014F49 66C0C0C0663C00 <1>
159 00014F50 F86C6666666CF800FE- <1> db 0f8h, 06ch, 066h, 066h, 066h, 06ch, 0f8h, 000h, 0feh, 062h, 068h, 078h, 068h, 062h, 0feh, 000h
159 00014F59 6268786862FE00 <1>
```

160	00014F60	FE6268786860F0003C-	<1>	db	0feh, 062h, 068h, 078h, 068h, 060h, 0f0h, 000h, 03ch, 066h, 0c0h, 0c0h, 0ceh, 066h, 03eh, 000h
160	00014F69	66C0C0CE663E00	<1>	db	0cch, 0cch, 0cch, 0fch, 0cch, 0cch, 0cch, 000h, 078h, 030h, 030h, 030h, 030h, 030h, 078h, 000h
161	00014F70	CCCCCFCCCCCCC0078-	<1>	db	01eh, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 0e6h, 066h, 06ch, 078h, 06ch, 066h, 0e6h, 000h
161	00014F79	30303030307800	<1>	db	0f0h, 060h, 060h, 060h, 062h, 066h, 0feh, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 000h
162	00014F80	1E0C0C0CCCCC7800E6-	<1>	db	0c6h, 0e6h, 0f6h, 0deh, 0ceh, 0c6h, 0c6h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h
162	00014F89	666C786C66E600	<1>	db	0fch, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 078h, 0cch, 0cch, 0cch, 0dch, 078h, 01ch, 000h
163	00014F90	F06060606266FE00C6-	<1>	db	0fch, 0b4h, 030h, 030h, 030h, 030h, 078h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 0fch, 000h
163	00014F99	EEFEFED6C6C600	<1>	db	0cch, 0cch, 0cch, 0cch, 0cch, 078h, 030h, 000h, 0c6h, 0c6h, 0c6h, 0d6h, 0feh, 0eeh, 0c6h, 000h
164	00014FA0	C6E6F6DECEC6C60038-	<1>	db	0c6h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 030h, 078h, 000h
164	00014FA9	6CC6C6C66C3800	<1>	db	0feh, 0c6h, 08ch, 018h, 032h, 066h, 0feh, 000h, 078h, 060h, 060h, 060h, 060h, 060h, 078h, 000h
165	00014FB0	FC66667C6060F00078-	<1>	db	0c0h, 060h, 030h, 018h, 00ch, 006h, 002h, 000h, 078h, 018h, 018h, 018h, 018h, 018h, 078h, 000h
165	00014FB9	CCCCCDC781C00	<1>	db	010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh
166	00014FC0	FC66667C6C66E60078-	<1>	db	030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 076h, 000h
166	00014FC9	CCE0701CCC7800	<1>	db	0e0h, 060h, 060h, 07ch, 066h, 066h, 0dch, 000h, 000h, 000h, 078h, 0cch, 0c0h, 0cch, 078h, 000h
167	00014FD0	FCB4303030307800CC-	<1>	db	01ch, 00ch, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
167	00014FD9	CCCCCCCCCF00	<1>	db	038h, 06ch, 060h, 0f0h, 060h, 060h, 0f0h, 000h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 0f8h
168	00014FE0	CCCCCCCCC783000C6-	<1>	db	0e0h, 060h, 06ch, 076h, 066h, 066h, 0e6h, 000h, 030h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
168	00014FE9	C6C6D6FEEEC600	<1>	db	00ch, 000h, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 0e0h, 060h, 066h, 06ch, 078h, 06ch, 0e6h, 000h
169	00014FF0	C6C66C38386CC600CC-	<1>	db	070h, 030h, 030h, 030h, 030h, 030h, 078h, 000h, 000h, 000h, 0cch, 0feh, 0feh, 0d6h, 0c6h, 000h
169	00014FF9	CCCC7830307800	<1>	db	000h, 000h, 0f8h, 0cch, 0cch, 0cch, 0cch, 000h, 000h, 000h, 078h, 0cch, 0cch, 0cch, 078h, 000h
170	00015000	FEC68C183266FE0078-	<1>	db	000h, 000h, 0dch, 066h, 066h, 07ch, 060h, 0f0h, 000h, 000h, 076h, 0cch, 0cch, 07ch, 00ch, 01eh
170	00015009	60606060607800	<1>	db	000h, 000h, 0dch, 076h, 066h, 060h, 0f0h, 000h, 000h, 000h, 07ch, 0c0h, 078h, 00ch, 0f8h, 000h
171	00015010	C06030180C06020078-	<1>	db	010h, 030h, 07ch, 030h, 030h, 034h, 018h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 076h, 000h
171	00015019	18181818187800	<1>	db	000h, 000h, 0cch, 0cch, 0cch, 078h, 030h, 000h, 000h, 000h, 0c6h, 0d6h, 0feh, 0feh, 06ch, 000h
172	00015020	10386CC60000000000-	<1>	db	000h, 000h, 0c6h, 06ch, 038h, 06ch, 0c6h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 07ch, 00ch, 0f8h
172	00015029	000000000000FF	<1>	db	000h, 000h, 0fch, 098h, 030h, 064h, 0fch, 000h, 01ch, 030h, 030h, 0e0h, 030h, 030h, 01ch, 000h
173	00015030	303018000000000000-	<1>	db	018h, 018h, 018h, 000h, 018h, 018h, 018h, 000h, 0e0h, 030h, 030h, 01ch, 030h, 030h, 0e0h, 000h
173	00015039	00780C7CCC7600	<1>	db	076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h
174	00015040	E060607C6666DC0000-	<1>	db	078h, 0cch, 0c0h, 0cch, 078h, 018h, 00ch, 078h, 000h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
174	00015049	0078CCC0CC7800	<1>	db	01ch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 07eh, 0c3h, 03ch, 006h, 03eh, 066h, 03fh, 000h
175	00015050	1C0C0C7CCCC760000-	<1>	db	0cch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 0e0h, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h
175	00015059	0078CCFCC07800	<1>	db	030h, 030h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 000h, 000h, 078h, 0c0h, 0c0h, 078h, 00ch, 038h
176	00015060	386C60F06060F00000-	<1>	db	07eh, 0c3h, 03ch, 066h, 07eh, 060h, 03ch, 000h, 0cch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
176	00015069	0076CCCC7C0CF8	<1>	db	0e0h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 0cch, 000h, 070h, 030h, 030h, 030h, 078h, 000h
177	00015070	E0606C766666E60030-	<1>	db	0c6h, 038h, 06ch, 0c6h, 0feh, 0c6h, 0c6h, 000h, 030h, 030h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
177	00015079	00703030307800	<1>	db	000h, 000h, 0c6h, 06ch, 038h, 06ch, 0c6h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 07ch, 00ch, 0f8h
178	00015080	0C000C0C0CCCC78E0-	<1>	db	000h, 000h, 0fch, 098h, 030h, 064h, 0fch, 000h, 01ch, 030h, 030h, 0e0h, 030h, 030h, 01ch, 000h
178	00015089	60666C786CE600	<1>	db	018h, 018h, 018h, 000h, 018h, 018h, 018h, 000h, 0e0h, 030h, 030h, 01ch, 030h, 030h, 0e0h, 000h
179	00015090	703030303030780000-	<1>	db	076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h
179	00015099	00CCFEFED6C600	<1>	db	078h, 0cch, 0c0h, 0cch, 078h, 018h, 00ch, 078h, 000h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
180	000150A0	0000F8CCCCCCCC0000-	<1>	db	01ch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 07eh, 0c3h, 03ch, 006h, 03eh, 066h, 03fh, 000h
180	000150A9	0078CCCCC7800	<1>	db	0cch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 0e0h, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h
181	000150B0	0000DC66667C60F000-	<1>	db	030h, 030h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 000h, 000h, 078h, 0c0h, 0c0h, 078h, 00ch, 038h
181	000150B9	0076CCCC7C0C1E	<1>	db	07eh, 0c3h, 03ch, 066h, 07eh, 060h, 03ch, 000h, 0cch, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
182	000150C0	0000DC766660F00000-	<1>	db	0e0h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h, 0cch, 000h, 070h, 030h, 030h, 030h, 078h, 000h
182	000150C9	007CC0780CF800	<1>	db	07ch, 0c6h, 038h, 018h, 018h, 018h, 03ch, 000h, 0e0h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
183	000150D0	10307C303034180000-	<1>	db	0c6h, 038h, 06ch, 0c6h, 0feh, 0c6h, 0c6h, 000h, 030h, 030h, 000h, 078h, 0cch, 0fch, 0c0h, 078h, 000h
183	000150D9	00CCCCCCCC7600	<1>	db	01ch, 000h, 0fch, 060h, 078h, 060h, 0fch, 000h, 000h, 000h, 07fh, 00ch, 07fh, 0cch, 07fh, 000h
184	000150E0	0000CCCCC783000000-	<1>	db	03eh, 06ch, 0cch, 0feh, 0cch, 0cch, 0ceh, 000h, 078h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h
184	000150E9	00C6D6FEFE6C00	<1>	db	000h, 0cch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 0e0h, 000h, 078h, 0cch, 0cch, 078h, 000h
185	000150F0	0000C66C386CC60000-	<1>	db	078h, 0cch, 000h, 0cch, 0cch, 0cch, 07eh, 000h, 000h, 0e0h, 000h, 0cch, 0cch, 0cch, 07eh, 000h
185	000150F9	00CCCCC7C0CF8	<1>	db	000h, 0cch, 000h, 0cch, 0cch, 07ch, 00ch, 0f8h, 0c3h, 018h, 03ch, 066h, 066h, 03ch, 018h, 000h
186	00015100	0000FC983064FC001C-	<1>	db	0cch, 000h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 018h, 018h, 07eh, 0c0h, 0c0h, 07eh, 018h, 018h
186	00015109	3030E030301C00	<1>	db	038h, 06ch, 064h, 0f0h, 060h, 0e6h, 0fch, 000h, 0cch, 0cch, 078h, 0fch, 030h, 0fch, 030h, 030h
187	00015110	1818180018181800E0-	<1>	db	0f8h, 0cch, 0cch, 0fah, 0c6h, 0cfh, 0c6h, 0c7h, 00eh, 01bh, 018h, 03ch, 018h, 018h, 0d8h, 070h
187	00015119	30301C3030E000	<1>	db	01ch, 000h, 078h, 00ch, 07ch, 0cch, 07eh, 000h, 038h, 000h, 070h, 030h, 030h, 030h, 078h, 000h
188	00015120	76DC00000000000000-	<1>	db	000h, 01ch, 000h, 078h, 0cch, 0cch, 078h, 000h, 000h, 01ch, 000h, 0cch, 0cch, 0cch, 07eh, 000h
188	00015129	10386CC6C6FE00	<1>	db	000h, 0f8h, 000h, 0f8h, 0cch, 0cch, 0cch, 000h, 0fch, 000h, 0cch, 0ech, 0fch, 0dch, 0cch, 000h
189	00015130	78CCC0CC78180C7800-	<1>	db	03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h
189	00015139	CC00CCCCC7E00	<1>	db	030h, 000h, 030h, 060h, 0c0h, 0cch, 078h, 000h, 000h, 000h, 000h, 0fch, 0c0h, 0c0h, 000h, 000h
190	00015140	1C0078CCFC078007E-	<1>	db	000h, 000h, 000h, 0fch, 00ch, 00ch, 000h, 000h, 0c3h, 0c6h, 0cch, 0deh, 033h, 066h, 0cch, 00fh
190	00015149	C33C063E663F00	<1>	db	0c3h, 0c6h, 0cch, 0dbh, 037h, 06fh, 0cfh, 003h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 000h
191	00015150	CC00780C7CCC7E00E0-	<1>	db	000h, 033h, 066h, 0cch, 066h, 033h, 000h, 000h, 000h, 0cch, 066h, 033h, 066h, 0cch, 000h, 000h
191	00015159	00780C7CCC7E00	<1>	db	022h, 088h, 022h, 088h, 022h, 088h, 022h, 088h, 055h, 0aah, 055h, 0aah, 055h, 0aah, 055h, 0aah
192	00015160	3030780C7CCC7E0000-	<1>	db	0dbh, 077h, 0dbh, 0eeh, 0dbh, 077h, 0dbh, 0eeh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
192	00015169	0078C0C0780C38	<1>	db	018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h
193	00015170	7EC33C667E603C00CC-	<1>	db	036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h
193	00015179	0078CCFCC07800	<1>	db	000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h
194	00015180	E00078CCFC07800CC-	<1>	db	036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h
194	00015189	00703030307800	<1>	db	036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h
195	00015190	7CC6381818183C00E0-	<1>	db	018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h
195	00015199	00703030307800	<1>	db	036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h
196	000151A0	C6386CC6FEC6C60030-	<1>	db	000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h
196	000151A9	300078CCFCCC00	<1>	db	036h, 036h, 036h, 036h, 036h, 036h, 036h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h
197	000151B0	1C00FC607860FC0000-	<1>	db	036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h
197	000151B9	007F0C7FCC7F00	<1>	db	018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h
198	000151C0	3E6CCCFECCCCCE0078-	<1>	db	036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h
198	000151C9	CC0078CCCC7800	<1>	db	000h, 000h, 000h, 0fch, 00ch, 00ch, 000h, 000h, 0c3h, 0c6h, 0cch, 0deh, 033h, 066h, 0cch, 00fh
199	000151D0	00CC0078CCCC780000-	<1>	db	0c3h, 0c6h, 0cch, 0dbh, 037h, 06fh, 0cfh, 003h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 000h
199	000151D9	E00078CCCC7800	<1>	db	000h, 033h, 066h, 0cch, 066h, 033h, 000h, 000h, 000h, 0cch, 066h, 033h, 066h, 0cch, 000h, 000h
200	000151E0	78CC00CCCCC7E0000-	<1>	db	022h, 088h, 022h, 088h, 022h, 088h, 022h, 088h, 055h, 0aah, 055h, 0aah, 055

222	00015340	00000000FF18181818-	<1>	db	000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h
222	00015349	1818181F18181818	<1>		
223	00015350	00000000FF00000018-	<1>	db	000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h
223	00015359	181818FF18181818	<1>		
224	00015360	18181F181F18181836-	<1>	db	018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h
224	00015369	36363637363636	<1>		
225	00015370	363637303F00000000-	<1>	db	036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h
225	00015379	003F303736363636	<1>		
226	00015380	3636F700FF00000000-	<1>	db	036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h
226	00015389	00FF00F736363636	<1>		
227	00015390	363637303736363600-	<1>	db	036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h
227	00015399	00FF00FF00000000	<1>		
228	000153A0	3636F700F736363618-	<1>	db	036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h
228	000153A9	18FF00FF00000000	<1>		
229	000153B0	36363636FF00000000-	<1>	db	036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h
229	000153B9	00FF00FF18181818	<1>		
230	000153C0	00000000FF36363636-	<1>	db	000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h
230	000153C9	3636363F00000000	<1>		
231	000153D0	18181F181F00000000-	<1>	db	018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h
231	000153D9	001F181F18181818	<1>		
232	000153E0	000000003F36363636-	<1>	db	000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h
232	000153E9	363636FF36363636	<1>		
233	000153F0	1818FF18FF18181818-	<1>	db	018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h
233	000153F9	181818F800000000	<1>		
234	00015400	000000001F181818FF-	<1>	db	000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
234	00015409	FFFFFFFFFFFFFFFF	<1>		
235	00015410	00000000FFFFFFFFF0-	<1>	db	000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
235	00015419	F0F0F0F0F0F0F0	<1>		
236	00015420	0F0F0F0F0F0F0FFF-	<1>	db	00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h
236	00015429	FFFFFFFF0000000000	<1>		
237	00015430	000076DCC8DC760000-	<1>	db	000h, 000h, 076h, 0dch, 0c8h, 0dch, 076h, 000h, 000h, 078h, 0cch, 0f8h, 0cch, 0f8h, 0c0h, 0c0h
237	00015439	78CCF8CCF8C0C0	<1>		
238	00015440	00FCCCC0C0C0C00000-	<1>	db	000h, 0fch, 0cch, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 000h
238	00015449	FE6C6C6C6C6C00	<1>		
239	00015450	FCCC603060CCFC0000-	<1>	db	0fch, 0cch, 060h, 030h, 060h, 0cch, 0fch, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 070h, 000h
239	00015459	007ED8D8D87000	<1>		
240	00015460	00666666667C60C000-	<1>	db	000h, 066h, 066h, 066h, 066h, 07ch, 060h, 0c0h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 000h
240	00015469	76DC1818181800	<1>		
241	00015470	FC3078CCCC7830FC38-	<1>	db	0fch, 030h, 078h, 0cch, 0cch, 078h, 030h, 0fch, 038h, 06ch, 0c6h, 0feh, 0c6h, 06ch, 038h, 000h
241	00015479	6CC6FEC66C3800	<1>		
242	00015480	386CC6C66C6CEE001C-	<1>	db	038h, 06ch, 0c6h, 0c6h, 06ch, 06ch, 0eeh, 000h, 01ch, 030h, 018h, 07ch, 0cch, 0cch, 078h, 000h
242	00015489	30187CCCC7800	<1>		
243	00015490	00007EDBDB7E000006-	<1>	db	000h, 000h, 07eh, 0dbh, 0dbh, 07eh, 000h, 000h, 006h, 00ch, 07eh, 0dbh, 0dbh, 07eh, 060h, 0c0h
243	00015499	0C7EDBDB7E60C0	<1>		
244	000154A0	3860C0F8C060380078-	<1>	db	038h, 060h, 0c0h, 0f8h, 0c0h, 060h, 038h, 000h, 078h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 000h
244	000154A9	CCCCCCCCCCCC00	<1>		
245	000154B0	00FC00FC00FC000030-	<1>	db	000h, 0fch, 000h, 0fch, 000h, 0fch, 000h, 000h, 030h, 030h, 0fch, 030h, 030h, 000h, 0fch, 000h
245	000154B9	30FC303000FC00	<1>		
246	000154C0	603018306000FC0018-	<1>	db	060h, 030h, 018h, 030h, 060h, 000h, 0fch, 000h, 018h, 030h, 060h, 030h, 018h, 000h, 0fch, 000h
246	000154C9	3060301800FC00	<1>		
247	000154D0	0E1B18181818181818-	<1>	db	00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 070h
247	000154D9	18181818D8D870	<1>		
248	000154E0	303000FC0030300000-	<1>	db	030h, 030h, 000h, 0fch, 000h, 030h, 030h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h
248	000154E9	76DC0076DC0000	<1>		
249	000154F0	386C6C380000000000-	<1>	db	038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h
249	000154F9	0000181800000000	<1>		
250	00015500	00000000180000000F-	<1>	db	000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 0ech, 06ch, 03ch, 01ch
250	00015509	0C0C0CEC6C3C1C	<1>		
251	00015510	786C6C6C6C00000070-	<1>	db	078h, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 070h, 018h, 030h, 060h, 078h, 000h, 000h, 000h
251	00015519	1830607800000000	<1>		
252	00015520	00003C3C3C3C000000-	<1>	db	000h, 000h, 03ch, 03ch, 03ch, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
252	00015529	0000000000000000	<1>		
253			<1>	vgaFont14:	
254	00015530	000000000000000000-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
254	00015539	0000000000000000	<1>		
255	00015540	7E81A58181BD99817E-	<1>	db	07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 07eh, 000h, 000h, 000h, 000h, 000h, 07eh, 0ffh
255	00015549	00000000007EFF	<1>		
256	00015550	DBFFFFC3E7FF7E0000-	<1>	db	0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 06ch, 0feh, 0feh
256	00015559	0000000006CFEFE	<1>		
257	00015560	FEFE7C381000000000-	<1>	db	0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch
257	00015569	000010387CFE7C	<1>		
258	00015570	381000000000000018-	<1>	db	038h, 010h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h
258	00015579	3C3CE7E7E71818	<1>		
259	00015580	3C0000000000183C7E-	<1>	db	03ch, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h
259	00015589	FFFF7E18183C00	<1>		
260	00015590	00000000000000183C-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h
260	00015599	3C18000000000000	<1>		
261	000155A0	FFFFFFFFF7C3C3E7-	<1>	db	0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h
261	000155A9	FFFFFFFFF0000	<1>		
262	000155B0	00003C664242663C00-	<1>	db	000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh
262	000155B9	000000FFFFFFFF	<1>		
263	000155C0	C399BDBD99C3FFFFFF-	<1>	db	0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 01eh, 00eh, 01ah, 032h
263	000155C9	FF00001E0E1A32	<1>		
264	000155D0	78CCCCC78000000000-	<1>	db	078h, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 066h, 066h, 03ch, 018h
264	000155D9	003C66666663C18	<1>		
265	000155E0	7E181800000000003F-	<1>	db	07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 070h, 0f0h
265	000155E9	333F30303070F0	<1>		
266	000155F0	E000000000007F637F-	<1>	db	0e0h, 000h, 000h, 000h, 000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h
266	000155F9	63636367E7E6C0	<1>		
267	00015600	000000001818DB3CE7-	<1>	db	000h, 000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h
267	00015609	3CDB181800000000	<1>		
268	00015610	000080C0E0F8FEF8E0-	<1>	db	000h, 000h, 080h, 0c0h, 0e0h, 0f8h, 0feh, 0f8h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h, 000h
268	00015619	C080000000000000	<1>		
269	00015620	02060E3EFE3E0E0602-	<1>	db	002h, 006h, 00eh, 03eh, 0feh, 03eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch
269	00015629	0000000000183C	<1>		
270	00015630	7E1818187E3C180000-	<1>	db	07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h
270	00015639	0000006666666666	<1>		
271	00015640	666600666600000000-	<1>	db	066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh
271	00015649	007FDBDBDB7B1B	<1>		
272	00015650	1B1B18000000007CC6-	<1>	db	01bh, 01bh, 01bh, 000h, 000h, 000h, 000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h
272	00015659	60386CC6C66C38	<1>		
273	00015660	0CC67C000000000000-	<1>	db	00ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 000h
273	00015669	000000FEFEFE00	<1>		
274	00015670	00000000183C7E1818-	<1>	db	000h, 000h, 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h
274	00015679	187E3C187E0000	<1>		
275	00015680	0000183C7E18181818-	<1>	db	000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
275	00015689	1818000000000000	<1>		
276	00015690	1818181818187E3C18-	<1>	db	018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
276	00015699	0000000000000000	<1>		
277	000156A0	180CFE0C1800000000-	<1>	db	018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 030h, 060h
277	000156A9	00000000003060	<1>		
278	000156B0	FE6030000000000000-	<1>	db	0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h
278	000156B9	00000000C0C0C0	<1>		
279	000156C0	FE0000000000000000-	<1>	db	0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 028h, 06ch, 0feh, 06ch, 028h, 000

284	00015719	0000006C6CFE6C	<1>	db	06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h
285	00015720	6C6CFE6C6C00000018-	<1>		
285	00015729	187CC6C2C07C06	<1>	db	086h, 0c6h, 07ch, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 066h
286	00015730	86C67C181800000000-	<1>		
286	00015739	00C2C60C183066	<1>	db	0c6h, 000h, 000h, 000h, 000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 076h, 000h
287	00015740	C60000000000386C6C-	<1>		
287	00015749	3876DCCCC7600	<1>	db	000h, 000h, 000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
288	00015750	000000303030600000-	<1>		
288	00015759	0000000000000000	<1>	db	000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h, 000h
289	00015760	00000C183030303030-	<1>		
289	00015769	180C0000000000	<1>	db	030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
290	00015770	30180C0C0C0C0C1830-	<1>		
290	00015779	0000000000000000	<1>	db	066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h
291	00015780	663CFF3C6600000000-	<1>		
291	00015789	00000000001818	<1>	db	07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
292	00015790	7E1818000000000000-	<1>		
292	00015799	0000000000000000	<1>	db	018h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h, 000h
293	000157A0	181818300000000000-	<1>		
293	000157A9	000000FE000000	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h
294	000157B0	000000000000000000-	<1>		
294	000157B9	00000000181800	<1>	db	000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h
295	000157C0	0000000002060C1830-	<1>		
295	000157C9	60C08000000000	<1>	db	000h, 000h, 07ch, 0c6h, 0ceh, 0deh, 0f6h, 0e6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
296	000157D0	00007CC6CEDEF6E6C6-	<1>		
296	000157D9	C67C000000000000	<1>	db	018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h
297	000157E0	18387818181818187E-	<1>		
297	000157E9	00000000007CC6	<1>	db	006h, 00ch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 006h, 006h
298	000157F0	060C183060C6FE0000-	<1>		
298	000157F9	0000007CC60606	<1>	db	03ch, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh
299	00015800	3C0606C67C00000000-	<1>		
299	00015809	000C1C3C6CCCFE	<1>	db	00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 0c6h
300	00015810	0C0C1E0000000000FE-	<1>		
300	00015819	C0C0C0FC0606C6	<1>	db	07ch, 000h, 000h, 000h, 000h, 000h, 038h, 060h, 0c0h, 0c0h, 0fch, 0c6h, 0c6h, 0c6h, 07ch, 000h
301	00015820	7C00000000003860C0-	<1>		
301	00015829	C0FCC6C6C67C00	<1>	db	000h, 000h, 000h, 000h, 0feh, 0c6h, 006h, 00ch, 018h, 030h, 030h, 030h, 030h, 000h, 000h, 000h
302	00015830	00000000FEC6060C18-	<1>		
302	00015839	3030303000000000	<1>	db	000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
303	00015840	00007CC6C6C67CC6C6-	<1>		
303	00015849	C67C000000000000	<1>	db	07ch, 0c6h, 0c6h, 0c6h, 07eh, 006h, 006h, 00ch, 078h, 000h, 000h, 000h, 000h, 000h, 000h, 018h
304	00015850	7CC6C6C67E06060C78-	<1>		
304	00015859	0000000000000018	<1>	db	018h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h
305	00015860	180000001818000000-	<1>		
305	00015869	00000000181800	<1>	db	000h, 000h, 018h, 018h, 030h, 000h, 000h, 000h, 000h, 000h, 000h, 006h, 00ch, 018h, 030h, 060h, 030h
306	00015870	000018183000000000-	<1>		
306	00015879	00060C18306030	<1>	db	018h, 00ch, 006h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07eh, 000h, 000h, 07eh, 000h
307	00015880	180C06000000000000-	<1>		
307	00015889	00007E00007E00	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 060h, 000h
308	00015890	000000000000603018-	<1>		
308	00015899	0C060C18306000	<1>	db	000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 00ch, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h
309	000158A0	000000007CC6C60C18-	<1>		
309	000158A9	18001818000000	<1>	db	000h, 000h, 07ch, 0c6h, 0c6h, 0deh, 0deh, 0deh, 0dch, 0c0h, 07ch, 000h, 000h, 000h, 000h, 000h
310	000158B0	00007CC6C6DEDEDC-	<1>		
310	000158B9	C07C000000000000	<1>	db	010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h, 0fch, 066h
311	000158C0	10386CC6C6FEC6C6C6-	<1>		
311	000158C9	0000000000FC66	<1>	db	066h, 066h, 07ch, 066h, 066h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h
312	000158D0	66667C666666FC0000-	<1>		
312	000158D9	0000003C66C2C0	<1>	db	0c0h, 0c0h, 0c2h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h, 0feh, 06ch, 066h, 066h, 066h, 066h
313	000158E0	C0C0C2663C00000000-	<1>		
313	000158E9	00F86C66666666	<1>	db	066h, 06ch, 0f8h, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 062h, 066h
314	000158F0	666C800000000000FE-	<1>		
314	000158F9	66626878686266	<1>	db	0feh, 000h, 000h, 000h, 000h, 000h, 0feh, 066h, 062h, 068h, 078h, 068h, 060h, 060h, 0f0h, 000h
315	00015900	FE0000000000FE6662-	<1>		
315	00015909	6878686060F000	<1>	db	000h, 000h, 000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0deh, 0c6h, 066h, 03ah, 000h, 000h, 000h
316	00015910	000000003C66C2C0C0-	<1>		
316	00015919	DEC6663A000000	<1>	db	000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h
317	00015920	0000C6C6C6C6FEC6C6-	<1>		
317	00015929	C6C6000000000000	<1>	db	03ch, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 01eh, 00ch
318	00015930	3C181818181818183C-	<1>		
318	00015939	000000000001E0C	<1>	db	00ch, 00ch, 00ch, 00ch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h, 000h, 0e6h, 066h, 06ch, 06ch
319	00015940	0C0C0C0CCCC780000-	<1>		
319	00015949	000000E6666C6C	<1>	db	078h, 06ch, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 0f0h, 060h, 060h, 060h, 060h, 060h
320	00015950	786C6C66E600000000-	<1>		
320	00015959	00F06060606060	<1>	db	062h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 0c6h, 0eeh, 0feh, 0feh, 0d6h, 0c6h, 0c6h, 0c6h
321	00015960	6266FE0000000000C6-	<1>		
321	00015969	EEFEFED6C6C6C6	<1>	db	0c6h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h
322	00015970	C6000000000000C6E6F6-	<1>		
322	00015979	FEDECEC6C6C600	<1>	db	000h, 000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h
323	00015980	00000000386CC6C6C6-	<1>		
323	00015989	C6C66C38000000	<1>	db	000h, 000h, 0feh, 066h, 066h, 066h, 07ch, 060h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h
324	00015990	0000FC6666667C6060-	<1>		
324	00015999	60F0000000000000	<1>	db	07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0deh, 07ch, 00ch, 00eh, 000h, 000h, 000h, 0feh, 066h
325	000159A0	7CC6C6C6C6D6DE7C0C-	<1>		
325	000159A9	0E00000000FC66	<1>	db	066h, 066h, 07ch, 06ch, 066h, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 060h
326	000159B0	66667C6C6666E60000-	<1>		
326	000159B9	0000007CC6C660	<1>	db	038h, 00ch, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 07eh, 07eh, 05ah, 018h, 018h, 018h
327	000159C0	380CC6C67C00000000-	<1>		
327	000159C9	007E7E5A181818	<1>	db	018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h
328	000159D0	18183C00000000000C6-	<1>		
328	000159D9	C6C6C6C6C6C6C6	<1>	db	07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 010h, 000h
329	000159E0	7C0000000000C6C6C6-	<1>		
329	000159E9	C6C6C66C381000	<1>	db	000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 07ch, 06ch, 000h, 000h, 000h
330	000159F0	00000000C6C6C6D6-	<1>		
330	000159F9	D6FE7C6C000000	<1>	db	000h, 000h, 0c6h, 0c6h, 06ch, 038h, 038h, 038h, 06ch, 0c6h, 0c6h, 000h, 000h, 000h, 000h, 000h
331	00015A00	0000C6C66C3838386C-	<1>		
331	00015A09	C6C60000000000	<1>	db	066h, 066h, 066h, 066h, 03ch, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 0feh, 0c6h
332	00015A10	666666663C1818183C-	<1>		
332	00015A19	0000000000FEC6	<1>	db	08ch, 018h, 030h, 060h, 0c2h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h, 03ch, 030h, 030h, 030h
333	00015A20	8C183060C2C6FE0000-	<1>		
333	00015A29	0000003C303030	<1>	db	030h, 030h, 030h, 030h, 03ch, 000h, 000h, 000h, 000h, 000h, 080h, 0c0h, 0e0h, 070h, 038h, 01ch
334	00015A30	303030303C00000000-	<1>		
334	00015A39	0080C0E070381C	<1>	db	00eh, 006h, 002h, 000h, 000h, 000h, 000h, 000h, 03ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch
335	00015A40	0E060200000000003C-	<1>		
335	00015A49	0C0C0C0C0C0C0C	<1>	db	03ch, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
336	00015A50	3C00000010386CC600-	<1>		
336	00015A59	0000000000000000	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h
337	00015A60	000000000000000000-	<1>		
337	00015A69	0000000000FF00	<1>	db	030h, 030h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
338	00015A70	303018000000000000-	<1>		
338	00015A79	0000000000000000	<1>	db	000h, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 0e0h, 060h
339	00015A80	000000780C7CCCC76-	<1>		
339	00015A89	0000000000E060	<1>	db	060h, 078h, 06ch, 066h, 066h, 066h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch
340	00015A90	60786C6666667C0000-	<1>		
340	00015A99	00000000000007C	<1>	db	0c6h, 0c0h,

346	00015AF9	000000000000606	<1>	db	000h, 00eh, 006h, 006h, 006h, 006h, 006h, 006h, 006h, 03ch, 000h, 000h, 000h, 0e0h, 060h, 060h, 066h
347	00015B00	000E060606066663C-	<1>	db	06ch, 078h, 06ch, 066h, 0e6h, 000h, 000h, 000h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h
347	00015B09	000000E0606066	<1>		
348	00015B10	6C786C66E600000000-	<1>	db	018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ech, 0feh, 0d6h, 0d6h, 0d6h
348	00015B19	0038181818181818	<1>	db	0c6h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 000h
349	00015B20	18183C000000000000-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h
349	00015B29	0000ECFED6D6D6	<1>	db	000h, 000h, 000h, 000h, 000h, 0dch, 066h, 066h, 066h, 07ch, 060h, 060h, 0f0h, 000h, 000h, 000h
350	00015B30	C60000000000000000-	<1>	db	000h, 000h, 000h, 076h, 0cch, 0cch, 0cch, 07ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h, 000h
350	00015B39	DC6666666666600	<1>	db	000h, 0dch, 076h, 066h, 060h, 060h, 0f0h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch
351	00015B40	000000000000007CC6-	<1>	db	0c6h, 070h, 01ch, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h
351	00015B49	C6C6C67C000000	<1>	db	030h, 036h, 01ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch
352	00015B50	0000000000DC666666-	<1>	db	076h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 03ch, 018h, 000h
352	00015B59	7C6060F0000000	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0d6h, 0d6h, 0feh, 06ch, 000h, 000h, 000h
353	00015B60	00000076CCCCC7C0C-	<1>	db	000h, 000h, 000h, 000h, 000h, 0c6h, 06ch, 038h, 038h, 06ch, 0c6h, 000h, 000h, 000h, 000h
353	00015B69	0C1E0000000000	<1>	db	000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h, 000h, 000h, 000h, 000h
354	00015B70	00DC76666060F00000-	<1>	db	000h, 0feh, 0cch, 018h, 030h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h, 00eh, 018h, 018h, 018h
354	00015B79	00000000000007C	<1>	db	070h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h
355	00015B80	C6701CC67C00000000-	<1>	db	018h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h
355	00015B89	00103030FC3030	<1>	db	070h, 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h
356	00015B90	30361C000000000000-	<1>	db	000h, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h
356	00015B99	0000CCCCCCCCC	<1>	db	000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h, 000h
357	00015BA0	760000000000000000-	<1>	db	0cch, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 00ch, 018h, 030h
357	00015BA9	666666663C1800	<1>	db	000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 078h
358	00015BB0	00000000000000C6C6-	<1>	db	00ch, 07ch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 000h, 00ch, 0cch, 0cch, 000h, 078h, 00ch, 07ch
358	00015BB9	D6D6FE6C000000	<1>	db	0cch, 0cch, 076h, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch
359	00015BC0	0000000000C66C3838-	<1>	db	076h, 000h, 000h, 000h, 000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h
359	00015BC9	6CC60000000000	<1>	db	000h, 000h, 000h, 000h, 000h, 03ch, 066h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h
360	00015BD0	000000C6C6C6C67E06-	<1>	db	000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 060h, 030h, 018h
360	00015BD9	0CF80000000000	<1>	db	0cch, 0cch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h
361	00015BE0	00FECC183066FE0000-	<1>	db	000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 000h, 038h
361	00015BE9	0000000E181818	<1>	db	018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h
362	00015BF0	701818180E00000000-	<1>	db	018h, 018h, 03ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h
362	00015BF9	00181818180018	<1>	db	03ch, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h
363	00015C00	181818000000000070-	<1>	db	000h, 000h, 038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h, 000h, 000h
363	00015C09	1818180E181818	<1>	db	018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h
364	00015C10	700000000000076DC00-	<1>	db	000h, 000h, 0cch, 076h, 036h, 07eh, 0d8h, 0d8h, 06eh, 000h, 000h, 000h, 000h, 000h, 03eh, 06ch
364	00015C19	00000000000000	<1>	db	0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 07ch
365	00015C20	00000000000010386C-	<1>	db	0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h, 07ch, 0c6h, 0c6h
365	00015C29	C6C6FE00000000	<1>	db	0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h
366	00015C30	00003C66C2C0C0C266-	<1>	db	07ch, 000h, 000h, 000h, 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h
366	00015C39	3C0C067C000000	<1>	db	000h, 000h, 000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h
367	00015C40	CCCC00CCCCCCCCC76-	<1>	db	000h, 000h, 0c6h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h, 000h, 0c6h
367	00015C49	000000000C1830	<1>	db	0c6h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h
368	00015C50	007CC6FEC0C67C0000-	<1>	db	000h, 000h, 0cch, 076h, 036h, 07eh, 0d8h, 0d8h, 06eh, 000h, 000h, 000h, 000h, 000h, 03eh, 06ch
368	00015C59	000010386C0078	<1>	db	0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 07ch
369	00015C60	0C7CCCC76000000000-	<1>	db	0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h, 07ch, 0c6h, 0c6h
369	00015C69	00CCCC00780C7C	<1>	db	07ch, 000h, 000h, 000h, 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h
370	00015C70	CCCC76000000006030-	<1>	db	000h, 000h, 000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h
370	00015C79	1800780C7CCCC	<1>	db	000h, 000h, 0c6h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h, 000h, 0c6h
371	00015C80	7600000000386C3800-	<1>	db	018h, 000h, 000h, 000h, 000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 076h, 000h
371	00015C89	780C7CCCC7600	<1>	db	000h, 000h, 000h, 000h, 000h, 03ch, 066h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h
372	00015C90	0000000000003C6660-	<1>	db	000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h
372	00015C99	663C0C063C0000	<1>	db	0cch, 0cch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h
373	00015CA0	0010386C007CC6FEC0-	<1>	db	000h, 07ch, 0c6h, 0feh, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 066h, 066h, 000h, 038h
373	00015CA9	C67C0000000000	<1>	db	018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h
374	00015CB0	CCCC007CC6FEC0C67C-	<1>	db	018h, 018h, 03ch, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h
374	00015CB9	00000000603018	<1>	db	03ch, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h
375	00015CC0	007CC6FEC0C67C0000-	<1>	db	000h, 000h, 038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 000h, 000h, 000h
375	00015CC9	00000066660038	<1>	db	018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h
376	00015CD0	181818183C00000000-	<1>	db	000h, 000h, 0cch, 076h, 036h, 07eh, 0d8h, 0d8h, 06eh, 000h, 000h, 000h, 000h, 000h, 03eh, 06ch
376	00015CD9	183C6600381818	<1>	db	0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 07ch
377	00015CE0	18183C000000006030-	<1>	db	0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h, 07ch, 0c6h, 0c6h
377	00015CE9	18003818181818	<1>	db	0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h
378	00015CF0	3C00000000C6C61038-	<1>	db	07ch, 000h, 000h, 000h, 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h
378	00015CF9	6CC6C6FEC6C600	<1>	db	000h, 000h, 0c6h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h, 000h, 0c6h
379	00015D00	0000386C3800386CC6-	<1>	db	0c6h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h
379	00015D09	C6FEC6C6000000	<1>	db	018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 066h, 0feh, 000h, 000h, 000h, 000h, 000h
380	00015D10	18306000FE66607C60-	<1>	db	000h, 000h, 0cch, 076h, 036h, 07eh, 0d8h, 0d8h, 06eh, 000h, 000h, 000h, 000h, 000h, 03eh, 06ch
380	00015D19	66FE0000000000	<1>	db	0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h, 010h, 038h, 06ch, 000h, 07ch
381	00015D20	0000CC76367ED8D86E-	<1>	db	0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h, 07ch, 0c6h, 0c6h
381	00015D29	00000000003E6C	<1>	db	0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h
382	00015D30	CCCCFECCCCCCE0000-	<1>	db	07ch, 000h, 000h, 000h, 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h
382	00015D39	000010386C007C	<1>	db	000h, 000h, 0c6h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h, 000h, 0c6h
383	00015D40	C6C6C6C67C00000000-	<1>	db	0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h, 07ch, 0c6h, 0c6h
383	00015D49	00C6C6007CC6C6	<1>	db	0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h
384	00015D50	C6C67C000000006030-	<1>	db	07ch, 000h, 000h, 000h, 000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h
384	00015D59	18007CC6C6C6C6	<1>	db	000h, 000h, 000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h
385	00015D60	7C000000003078CC00-	<1>	db	000h, 000h, 0c6h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h, 000h, 0c6h
385	00015D69	CCCCCCCCC7600	<1>	db	0c6h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 000h
386	00015D70	00000060301800CCCC-	<1>	db	0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h, 018h, 018h, 03ch, 066h, 060h
386	00015D79	CCCCC76000000	<1>	db	060h, 066h, 03ch, 018h, 018h, 000h, 000h, 000h, 000h, 038h

408	00015ED9	441144114455AA	<1>
409	00015EE0	5A5A5A5A5A5A5A55-	<1>
409	00015EE9	AA5A5AAD77DD77	<1>
410	00015EF0	DD77DD77DD77DD77DD-	<1>
410	00015EF9	7718181818181818	<1>
411	00015F00	181818181818181818-	<1>
411	00015F09	18181818181818F8	<1>
412	00015F10	18181818181818181818-	<1>
412	00015F19	1818F818F8181818	<1>
413	00015F20	181818183636363636-	<1>
413	00015F29	3636F63636363636	<1>
414	00015F30	36360000000000000000-	<1>
414	00015F39	FE36363636363636	<1>
415	00015F40	0000000000F818F818-	<1>
415	00015F49	1818181818183636	<1>
416	00015F50	363636F606F6363636-	<1>
416	00015F59	3636363636363636	<1>
417	00015F60	363636363636363636-	<1>
417	00015F69	360000000000FF	<1>
418	00015F70	06F636363636363636-	<1>
418	00015F79	36363636F606FF	<1>
419	00015F80	000000000000363636-	<1>
419	00015F89	36363636FE0000	<1>
420	00015F90	0000000001818181818-	<1>
420	00015F99	F818F800000000	<1>
421	00015FA0	00000000000000000000-	<1>
421	00015FA9	F818181818181818	<1>
422	00015FB0	1818181818181818F00-	<1>
422	00015FB9	000000000001818	<1>
423	00015FC0	1818181818FF000000-	<1>
423	00015FC9	0000000000000000	<1>
424	00015FD0	000000FF181818181818-	<1>
424	00015FD9	1818181818181818	<1>
425	00015FE0	181F181818181818000-	<1>
425	00015FE9	00000000000000FF	<1>
426	00015FF0	00000000000018181818-	<1>
426	00015FF9	18181818FF1818	<1>
427	00016000	18181818181818181818-	<1>
427	00016009	1F181F1818181818	<1>
428	00016010	18183636363636363636-	<1>
428	00016019	3736363636363636	<1>
429	00016020	363636363637303F00-	<1>
429	00016029	0000000000000000	<1>
430	00016030	0000003F3037363636-	<1>
430	00016039	3636363636363636	<1>
431	00016040	36F700FF000000000000-	<1>
431	00016049	00000000000000FF	<1>
432	00016050	00F73636363636363636-	<1>
432	00016059	36363636373037	<1>
433	00016060	36363636363600000000-	<1>
433	00016069	0000FF00FF0000	<1>
434	00016070	00000000363636363636-	<1>
434	00016079	F700F736363636	<1>
435	00016080	3636181818181818FF00-	<1>
435	00016089	FF00000000000000	<1>
436	00016090	36363636363636FF00-	<1>
436	00016099	0000000000000000	<1>
437	000160A0	000000FF00FF18181818-	<1>
437	000160A9	1818180000000000	<1>
438	000160B0	000000FF363636363636-	<1>
438	000160B9	3636363636363636	<1>
439	000160C0	363F000000000000018-	<1>
439	000160C9	181818181F181F	<1>
440	000160D0	00000000000000000000-	<1>
440	000160D9	00001F181F1818	<1>
441	000160E0	18181818000000000000-	<1>
441	000160E9	00003F36363636	<1>
442	000160F0	36363636363636363636-	<1>
442	000160F9	FF36363636363636	<1>
443	00016100	1818181818FF18FF18-	<1>
443	00016109	1818181818181818	<1>
444	00016110	1818181818F8000000-	<1>
444	00016119	0000000000000000	<1>
445	00016120	00000001F181818181818-	

[illegible]

```
470 000162B9 76Dc0000000000 <1>
471 000162C0 00386C6C3800000000- <1> db 000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
471 000162C9 00000000000000 <1>
472 000162D0 000000001818000000- <1> db 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
472 000162D9 00000000000000 <1>
473 000162E0 000000180000000000- <1> db 000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 00fh, 00ch, 00ch, 00ch, 00ch
473 000162E9 00000F0C0C0C0C <1>
474 000162F0 0CEC6C3C1C00000000- <1> db 00ch, 0ech, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h
474 000162F9 D86C6C6C6C6C00 <1>
475 00016300 0000000000000070D8- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h
475 00016309 3060C8F8000000 <1>
476 00016310 00000000000000007C- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h
476 00016319 7C7C7C7C7C0000 <1>
477 00016320 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
477 00016329 00000000000000 <1>
478 <1>
479 00016330 000000000000000000- <1> vgaFont16: db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
479 00016339 00000000000000 <1>
480 00016340 00007E81A58181BD99- <1> db 000h, 000h, 07eh, 081h, 0a5h, 081h, 081h, 0bdh, 099h, 081h, 081h, 07eh, 000h, 000h, 000h, 000h
480 00016349 81817E00000000 <1>
481 00016350 00007EFFDBFFFC3E7- <1> db 000h, 000h, 07eh, 0ffh, 0dbh, 0ffh, 0ffh, 0c3h, 0e7h, 0ffh, 0ffh, 07eh, 000h, 000h, 000h, 000h
481 00016359 FFFF7E00000000 <1>
482 00016360 000000006CFEFEFEFE- <1> db 000h, 000h, 000h, 000h, 06ch, 0feh, 0feh, 0feh, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h
482 00016369 7C381000000000 <1>
483 00016370 0000000010387CFE7C- <1> db 000h, 000h, 000h, 000h, 010h, 038h, 07ch, 0feh, 07ch, 038h, 010h, 000h, 000h, 000h, 000h, 000h
483 00016379 38100000000000 <1>
484 00016380 000000183C3CE7E7E7- <1> db 000h, 000h, 000h, 018h, 03ch, 03ch, 0e7h, 0e7h, 0e7h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
484 00016389 18183C00000000 <1>
485 00016390 000000183C7EFFFF7E- <1> db 000h, 000h, 000h, 018h, 03ch, 07eh, 0ffh, 0ffh, 07eh, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
485 00016399 18183C00000000 <1>
486 000163A0 000000000000183C3C- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 018h, 03ch, 03ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h
486 000163A9 18000000000000 <1>
487 000163B0 FFFFFFFF7C3C3- <1> db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0e7h, 0c3h, 0c3h, 0e7h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
487 000163B9 E7FFFFFFF <1>
488 000163C0 00000000003C664242- <1> db 000h, 000h, 000h, 000h, 000h, 03ch, 066h, 042h, 042h, 066h, 03ch, 000h, 000h, 000h, 000h, 000h
488 000163C9 663C0000000000 <1>
489 000163D0 FFFFFFFF399BDBD- <1> db 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0c3h, 099h, 0bdh, 0bdh, 099h, 0c3h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
489 000163D9 99C3FFFFFFF <1>
490 000163E0 00001E0E1A3278CCCC- <1> db 000h, 000h, 01eh, 00eh, 01ah, 032h, 078h, 0cch, 0cch, 0cch, 0cch, 078h, 000h, 000h, 000h, 000h
490 000163E9 CCCC7800000000 <1>
491 000163F0 00003C6666666663C18- <1> db 000h, 000h, 03ch, 066h, 066h, 066h, 066h, 03ch, 018h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h
491 000163F9 7E181800000000 <1>
492 00016400 00003F33F30303030- <1> db 000h, 000h, 03fh, 033h, 03fh, 030h, 030h, 030h, 030h, 070h, 0f0h, 0e0h, 000h, 000h, 000h, 000h
492 00016409 70F0E000000000 <1>
493 00016410 00007F637F63636363- <1> db 000h, 000h, 07fh, 063h, 07fh, 063h, 063h, 063h, 063h, 067h, 0e7h, 0e6h, 0c0h, 000h, 000h, 000h
493 00016419 67E7E6C0000000 <1>
494 00016420 0000001818DB3CE73C- <1> db 000h, 000h, 000h, 018h, 018h, 0dbh, 03ch, 0e7h, 03ch, 0dbh, 018h, 018h, 000h, 000h, 000h, 000h
494 00016429 DB181800000000 <1>
495 00016430 0080C0E0F0F8FEF8F0- <1> db 000h, 080h, 0c0h, 0e0h, 0f0h, 0f8h, 0feh, 0f8h, 0f0h, 0e0h, 0c0h, 080h, 000h, 000h, 000h, 000h
495 00016439 E0C08000000000 <1>
496 00016440 0002060E1E3EFE3E1E- <1> db 000h, 002h, 006h, 00eh, 01eh, 03eh, 0feh, 03eh, 01eh, 00eh, 006h, 002h, 000h, 000h, 000h, 000h
496 00016449 0E060200000000 <1>
497 00016450 0000183C7E1818187E- <1> db 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h
497 00016459 3C180000000000 <1>
498 00016460 000066666666666666- <1> db 000h, 000h, 066h, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 066h, 066h, 000h, 000h, 000h, 000h
498 00016469 00666600000000 <1>
499 00016470 00007FDBDBDB7B1B18- <1> db 000h, 000h, 07fh, 0dbh, 0dbh, 0dbh, 07bh, 01bh, 01bh, 01bh, 01bh, 01bh, 000h, 000h, 000h, 000h
499 00016479 1B1B1800000000 <1>
500 00016480 007CC660386CC6C66C- <1> db 000h, 07ch, 0c6h, 060h, 038h, 06ch, 0c6h, 0c6h, 06ch, 038h, 00ch, 0c6h, 07ch, 000h, 000h, 000h
500 00016489 380CC67C000000 <1>
501 00016490 0000000000000000FE- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0feh, 0feh, 0feh, 000h, 000h, 000h, 000h
501 00016499 FEFEFE00000000 <1>
502 000164A0 0000183C7E1818187E- <1> db 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 07eh, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h
502 000164A9 3C187E00000000 <1>
503 000164B0 0000183C7E18181818- <1> db 000h, 000h, 018h, 03ch, 07eh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
503 000164B9 18181800000000 <1>
504 000164C0 000018181818181818- <1> db 000h, 000h, 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 03ch, 018h, 000h, 000h, 000h, 000h, 000h
504 000164C9 7E3C1800000000 <1>
505 000164D0 0000000000180CFE0C- <1> db 000h, 000h, 000h, 000h, 000h, 018h, 00ch, 0feh, 00ch, 018h, 000h, 000h, 000h, 000h, 000h, 000h
505 000164D9 18000000000000 <1>
506 000164E0 00000000003060FE60- <1> db 000h, 000h, 000h, 000h, 000h, 030h, 060h, 0feh, 060h, 030h, 000h, 000h, 000h, 000h, 000h, 000h
506 000164E9 30000000000000 <1>
507 000164F0 000000000000C0C0C0- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 0c0h, 0c0h, 0c0h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h
507 000164F9 FE000000000000 <1>
508 00016500 00000000002466FF66- <1> db 000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h
508 00016509 24000000000000 <1>
509 00016510 000000001038387C7C- <1> db 000h, 000h, 000h, 000h, 010h, 038h, 038h, 07ch, 07ch, 0feh, 0feh, 000h, 000h, 000h, 000h, 000h, 000h
509 00016519 FEFE0000000000 <1>
510 00016520 00000000FEFE7C7C38- <1> db 000h, 000h, 000h, 000h, 0feh, 0feh, 07ch, 07ch, 038h, 038h, 010h, 000h, 000h, 000h, 000h, 000h
510 00016529 38100000000000 <1>
511 00016530 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
511 00016539 00000000000000 <1>
512 00016540 0000183C3C3C181818- <1> db 000h, 000h, 018h, 03ch, 03ch, 03ch, 018h, 018h, 018h, 000h, 018h, 018h, 000h, 000h, 000h, 000h
512 00016549 00181800000000 <1>
513 00016550 006666662400000000- <1> db 000h, 066h, 066h, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
513 00016559 00000000000000 <1>
514 00016560 0000006C6CFE6C6C6C- <1> db 000h, 000h, 000h, 06ch, 06ch, 0feh, 06ch, 06ch, 06ch, 0feh, 06ch, 06ch, 000h, 000h, 000h, 000h
514 00016569 FE6C6C00000000 <1>
515 00016570 18187CC6C2C07C0606- <1> db 018h, 018h, 07ch, 0c6h, 0c2h, 0c0h, 07ch, 006h, 006h, 086h, 0c6h, 07ch, 018h, 018h, 000h, 000h
515 00016579 86C67C18180000 <1>
516 00016580 00000000C2C60C1830- <1> db 000h, 000h, 000h, 000h, 0c2h, 0c6h, 00ch, 018h, 030h, 060h, 0c6h, 086h, 000h, 000h, 000h, 000h
516 00016589 60C68600000000 <1>
517 00016590 0000386C6C3876DCCC- <1> db 000h, 000h, 038h, 06ch, 06ch, 038h, 076h, 0dch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
517 00016599 CCCC7600000000 <1>
518 000165A0 003030306000000000- <1> db 000h, 030h, 030h, 030h, 060h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
518 000165A9 00000000000000 <1>
519 000165B0 00000C183030303030- <1> db 000h, 000h, 00ch, 018h, 030h, 030h, 030h, 030h, 030h, 030h, 018h, 00ch, 000h, 000h, 000h, 000h
519 000165B9 30180C00000000 <1>
520 000165C0 000030180C0C0C0C0C- <1> db 000h, 000h, 030h, 018h, 00ch, 00ch, 00ch, 00ch, 00ch, 00ch, 018h, 030h, 000h, 000h, 000h, 000h
520 000165C9 0C183000000000 <1>
521 000165D0 0000000000663CFF3C- <1> db 000h, 000h, 000h, 000h, 000h, 066h, 03ch, 0ffh, 03ch, 066h, 000h, 000h, 000h, 000h, 000h, 000h
521 000165D9 66000000000000 <1>
522 000165E0 000000000018187E18- <1> db 000h, 000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h
522 000165E9 18000000000000 <1>
523 000165F0 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 018h, 030h, 000h, 000h, 000h, 000h
523 000165F9 18181830000000 <1>
524 00016600 00000000000000FE00- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
524 00016609 00000000000000 <1>
525 00016610 000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
525 00016619 00181800000000 <1>
526 00016620 00000000002060C1830- <1> db 000h, 000h, 000h, 000h, 002h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 080h, 000h, 000h, 000h, 000h
526 00016629 60C08000000000 <1>
527 00016630 00003C66C3C3DBDBC3- <1> db 000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h, 000h, 000h
527 00016639 C3663C00000000 <1>
528 00016640 000018387818181818- <1> db 000h, 000h, 018h, 038h, 078h, 018h, 018h, 018h, 018h, 018h, 018h, 07eh, 000h, 000h, 000h, 000h
528 00016649 18187E00000000 <1>
529 00016650 00007CC6060C183060- <1> db 000h, 000h, 07ch, 0c6h, 006h, 00ch, 018h, 030h, 060h, 0c0h, 0c6h, 0feh, 000h, 000h, 000h, 000h
529 00016659 C0C6FE00000000 <1>
530 00016660 00007CC606063C0606- <1> db 000h, 000h, 07ch, 0c6h, 006h, 006h, 03ch, 006h, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h
530 00016669 06C67C00000000 <1>
531 00016670 00000C1C3C6CCCFE0C- <1> db 000h, 000h, 00ch, 01ch, 03ch, 06ch, 0cch, 0feh, 00ch, 00ch, 00ch, 01eh, 000h, 000h, 000h, 000h
531 00016679 0C0C1E00000000 <1>
532 00016680 0000FEC0C0C0FC0606- <1> db 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0fch, 006h, 006h, 006h, 0c6h, 07ch, 000h, 000h, 000h, 000h
532 00016689 06C67C00000000 <1>
```



533	00016690	00003860C0C0FCC6C6-	<1>	db	000h,	000h,	038h,	060h,	0c0h,	0c0h,	0fch,	0c6h,	0c6h,	0c6h,	0c6h,	07ch,	000h,	000h,	000h,	000h
533	00016699	C6C67C000000000	<1>																	
534	000166A0	0000FEC606060C1830-	<1>	db	000h,	000h,	0feh,	0c6h,	006h,	006h,	00ch,	018h,	030h,	030h,	030h,	030h,	000h,	000h,	000h,	000h
534	000166A9	303030000000000	<1>																	
535	000166B0	00007CC6C6C67CC6C6-	<1>	db	000h,	000h,	07ch,	0c6h,	0c6h,	0c6h,	07ch,	0c6h,	0c6h,	0c6h,	0c6h,	07ch,	000h,	000h,	000h,	000h
535	000166B9	C6C67C000000000	<1>																	
536	000166C0	00007CC6C6C67E0606-	<1>	db	000h,	000h,	07ch,	0c6h,	0c6h,	0c6h,	07eh,	006h,	006h,	006h,	00ch,	078h,	000h,	000h,	000h,	000h
536	000166C9	060C78000000000	<1>																	
537	000166D0	000000001818000000-	<1>	db	000h,	000h,	000h,	000h,	018h,	018h,	000h,	000h,	000h,	018h,	018h,	000h,	000h,	000h,	000h,	000h
537	000166D9	181800000000000	<1>																	
538	000166E0	000000001818000000-	<1>	db	000h,	000h,	000h,	000h,	018h,	018h,	000h,	000h,	000h,	018h,	018h,	030h,	000h,	000h,	000h,	000h
538	000166E9	181830000000000	<1>																	
539	000166F0	000000060C18306030-	<1>	db	000h,	000h,	000h,	006h,	00ch,	018h,	030h,	060h,	030h,	018h,	00ch,	006h,	000h,	000h,	000h,	000h
539	000166F9	180C06000000000	<1>																	
540	00016700	00000000007E00007E-	<1>	db	000h,	000h,	000h,	000h,	000h,	07eh,	000h,	000h,	07eh,	000h,	000h,	000h,	000h,	000h,	000h,	000h
540	00016709	000000000000000	<1>																	
541	00016710	0000006030180C060C-	<1>	db	000h,	000h,	000h,	060h,	030h,	018h,	00ch,	006h,	00ch,	018h,	030h,	060h,	000h,	000h,	000h,	000h
541	00016719	183060000000000	<1>																	
542	00016720	00007CC6C60C181818-	<1>	db	000h,	000h,	007ch,	0c6h,	0c6h,	00ch,	018h,	018h,	018h,	000h,	018h,	018h,	000h,	000h,	000h,	000h
542	00016729	001818000000000	<1>																	
543	00016730	0000007CC6C6DEDEDE-	<1>	db	000h,	000h,	000h,	07ch,	0c6h,	0c6h,	0deh,	0deh,	0deh,	0dch,	0c0h,	07ch,	000h,	000h,	000h,	000h
543	00016739	DC07C0000000000	<1>																	
544	00016740	000010386CC6C6FEC6-	<1>	db	000h,	000h,	010h,	038h,	06ch,	0c6h,	0c6h,	0feh,	0c6h,	0c6h,	0c6h,	0c6h,	000h,	000h,	000h,	000h
544	00016749	C6C6C6000000000	<1>																	
545	00016750	0000FC6666667C6666-	<1>	db	000h,	000h,	0fch,	066h,	066h,	066h,	07ch,	066h,	066h,	066h,	066h,	0fch,	000h,	000h,	000h	

595	00016A70	0000103030FC303030-	<1>	db	000h, 000h, 010h, 030h, 030h, 0fch, 030h, 030h, 030h, 030h, 036h, 01ch, 000h, 000h, 000h, 000h
595	00016A79	30361c000000000	<1>		
596	00016A80	0000000000CCCCCCCC-	<1>	db	000h, 000h, 000h, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
596	00016A89	CCCC76000000000	<1>		
597	00016A90	0000000000c3c3c3-	<1>	db	000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h
597	00016A99	663c18000000000	<1>		
598	00016AA0	0000000000c3c3c3DB-	<1>	db	000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h
598	00016AA9	DBFF66000000000	<1>		
599	00016AB0	0000000000c3663c18-	<1>	db	000h, 000h, 000h, 000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h
599	00016AB9	3c66c3000000000	<1>		
600	00016AC0	0000000000c6c6c6c6-	<1>	db	000h, 000h, 000h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 0f8h, 000h
600	00016AC9	c6c67E060CF800	<1>		
601	00016AD0	0000000000FECC1830-	<1>	db	000h, 000h, 000h, 000h, 000h, 0feh, 0cch, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h
601	00016AD9	60c6FE000000000	<1>		
602	00016AE0	00000E181818701818-	<1>	db	000h, 000h, 00eh, 018h, 018h, 018h, 070h, 018h, 018h, 018h, 018h, 00eh, 000h, 000h, 000h, 000h
602	00016AE9	18180E000000000	<1>		
603	00016AF0	000018181818001818-	<1>	db	000h, 000h, 018h, 018h, 018h, 018h, 000h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
603	00016AF9	181818000000000	<1>		
604	00016B00	0000701818180E1818-	<1>	db	000h, 000h, 070h, 018h, 018h, 018h, 00eh, 018h, 018h, 018h, 018h, 070h, 000h, 000h, 000h, 000h
604	00016B09	181870000000000	<1>		
605	00016B10	000076Dc0000000000-	<1>	db	000h, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
605	00016B19	000000000000000	<1>		
606	00016B20	0000000010386CC6C6-	<1>	db	000h, 000h, 000h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0c6h, 0feh, 000h, 000h, 000h, 000h, 000h
606	00016B29	C6FE00000000000	<1>		
607	00016B30	00003c66c2c0c0c0c2-	<1>	db	000h, 000h, 03ch, 066h, 0c2h, 0c0h, 0c0h, 0c0h, 0c2h, 066h, 03ch, 00ch, 006h, 07ch, 000h, 000h
607	00016B39	663c0c067c0000	<1>		
608	00016B40	0000CC0000CCCCCCCC-	<1>	db	000h, 000h, 0cch, 000h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
608	00016B49	CCCC76000000000	<1>		
609	00016B50	000c1830007CC6FEC0-	<1>	db	000h, 00ch, 018h, 030h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
609	00016B59	c0c67c000000000	<1>		
610	00016B60	0010386c00780c7CCC-	<1>	db	000h, 010h, 038h, 06ch, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
610	00016B69	CCCC76000000000	<1>		
611	00016B70	0000CC00000780c7CCC-	<1>	db	000h, 000h, 0cch, 000h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
611	00016B79	CCCC76000000000	<1>		
612	00016B80	0060301800780c7CCC-	<1>	db	000h, 060h, 030h, 018h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
612	00016B89	CCCC76000000000	<1>		
613	00016B90	00386c3800780c7CCC-	<1>	db	000h, 038h, 06ch, 038h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
613	00016B99	CCCC76000000000	<1>		
614	00016BA0	000000003c66606066-	<1>	db	000h, 000h, 000h, 000h, 03ch, 066h, 060h, 060h, 066h, 03ch, 00ch, 006h, 03ch, 000h, 000h, 000h
614	00016BA9	3c0c063c0000000	<1>		
615	00016BB0	0010386c007CC6FEC0-	<1>	db	000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
615	00016BB9	c0c67c000000000	<1>		
616	00016BC0	0000C600007CC6FEC0-	<1>	db	000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
616	00016BC9	c0c67c000000000	<1>		
617	00016BD0	00603018007CC6FEC0-	<1>	db	000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0feh, 0c0h, 0c0h, 0c6h, 07ch, 000h, 000h, 000h, 000h
617	00016BD9	c0c67c000000000	<1>		
618	00016BE0	000066000038181818-	<1>	db	000h, 000h, 066h, 000h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
618	00016BE9	18183c000000000	<1>		
619	00016BF0	00183c660038181818-	<1>	db	000h, 018h, 03ch, 066h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
619	00016BF9	18183c000000000	<1>		
620	00016C00	006030180038181818-	<1>	db	000h, 060h, 030h, 018h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
620	00016C09	18183c000000000	<1>		
621	00016C10	00c60010386CC6C6FE-	<1>	db	000h, 0c6h, 000h, 010h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
621	00016C19	c6c6c6000000000	<1>		
622	00016C20	386c3800386CC6C6FE-	<1>	db	038h, 06ch, 038h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
622	00016C29	c6c6c6000000000	<1>		
623	00016C30	18306000FE66607C60-	<1>	db	018h, 030h, 060h, 000h, 0feh, 066h, 060h, 07ch, 060h, 060h, 066h, 0feh, 000h, 000h, 000h, 000h
623	00016C39	6066FE000000000	<1>		
624	00016C40	00000000006E3B1B7E-	<1>	db	000h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 000h
624	00016C49	D8DC77000000000	<1>		
625	00016C50	00003E6CCCCCFECCCC-	<1>	db	000h, 000h, 03eh, 06ch, 0cch, 0cch, 0feh, 0cch, 0cch, 0cch, 0cch, 0cch, 0ceh, 000h, 000h, 000h, 000h
625	00016C59	CCCCCE000000000	<1>		
626	00016C60	0010386c007CC6C6C6-	<1>	db	000h, 010h, 038h, 06ch, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
626	00016C69	c6c67c000000000	<1>		
627	00016C70	0000c600007CC6C6C6-	<1>	db	000h, 000h, 0c6h, 000h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
627	00016C79	c6c67c000000000	<1>		
628	00016C80	00603018007CC6C6C6-	<1>	db	000h, 060h, 030h, 018h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
628	00016C89	c6c67c000000000	<1>		
629	00016C90	003078CC00CCCCCCCC-	<1>	db	000h, 030h, 078h, 0cch, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h
629	00016C99	CCCC76000000000	<1>		
630	00016CA0	0060301800CCCCCCCC-	<1>	db	000h, 060h, 030h, 018h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
630	00016CA9	CCCC76000000000	<1>		
631	00016CB0	0000c60000c6c6c6c6-	<1>	db	000h, 000h, 0c6h, 000h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07eh, 006h, 00ch, 078h, 000h
631	00016CB9	c6c67E060c7800	<1>		
632	00016CC0	00c6007CC6C6C6C6C6-	<1>	db	000h, 0c6h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
632	00016CC9	c6c67c000000000	<1>		
633	00016CD0	00c600c6c6c6c6c6c6-	<1>	db	000h, 0c6h, 000h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
633	00016CD9	c6c67c000000000	<1>		
634	00016CE0	0018187EC3C0C0C0C3-	<1>	db	000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h
634	00016CE9	7E1818000000000	<1>		
635	00016CF0	00386c6460F0606060-	<1>	db	000h, 038h, 06ch, 064h, 060h, 0f0h, 060h, 060h, 060h, 060h, 0e6h, 0fch, 000h, 000h, 000h, 000h
635	00016CF9	60E6FC000000000	<1>		
636	00016D00	0000C3663C18FF18FF-	<1>	db	000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h
636	00016D09	181818000000000	<1>		
637	00016D10	00FC66667C62666F66-	<1>	db	000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 066h, 0f3h, 000h, 000h, 000h, 000h
637	00016D19	6666F3000000000	<1>		
638	00016D20	000E1B1818187E1818-	<1>	db	000h, 00eh, 01bh, 018h, 018h, 018h, 07eh, 018h, 018h, 018h, 018h, 018h, 0d8h, 070h, 000h, 000h
638	00016D29	181818D8700000	<1>		
639	00016D30	0018306000780c7CCC-	<1>	db	000h, 018h, 030h, 060h, 000h, 078h, 00ch, 07ch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
639	00016D39	CCCC76000000000	<1>		
640	00016D40	000c18300038181818-	<1>	db	000h, 00ch, 018h, 030h, 000h, 038h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h
640	00016D49	18183c000000000	<1>		
641	00016D50	00183060007CC6C6C6-	<1>	db	000h, 018h, 030h, 060h, 000h, 07ch, 0c6h, 0c6h, 0c6h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
641	00016D59	c6c67c000000000	<1>		
642	00016D60	0018306000CCCCCCCC-	<1>	db	000h, 018h, 030h, 060h, 000h, 0cch, 0cch, 0cch, 0cch, 0cch, 0cch, 076h, 000h, 000h, 000h, 000h
642	00016D69	CCCC76000000000	<1>		
643	00016D70	000076Dc00Dc666666-	<1>	db	000h, 000h, 076h, 0dch, 000h, 0dch, 066h, 066h, 066h, 066h, 066h, 066h, 000h, 000h, 000h, 000h
643	00016D79	666666000000000	<1>		
644	00016D80	76Dc00c6E6F6FEDECE-	<1>	db	076h, 0dch, 000h, 0c6h, 0e6h, 0f6h, 0feh, 0deh, 0ceh, 0c6h, 0c6h, 0c6h, 000h, 000h, 000h, 000h
644	00016D89	c6c6c6000000000	<1>		
645	00016D90	003c6c6c3E007E0000-	<1>	db	000h, 03ch, 06ch, 06ch, 03eh, 000h, 07eh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
645	00016D99	000000000000000	<1>		
646	00016DA0	00386C6c38007C0000-	<1>	db	000h, 038h, 06ch, 06ch, 038h, 000h, 07ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
646	00016DA9	000000000000000	<1>		
647	00016DB0	0000303000303060C0-	<1>	db	000h, 000h, 030h, 030h, 000h, 030h, 030h, 060h, 0c0h, 0c6h, 0c6h, 07ch, 000h, 000h, 000h, 000h
647	00016DB9	c6c67c000000000	<1>		
648	00016DC0	000000000000FEC0C0-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 0feh, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h, 000h
648	00016DC9	c0c000000000000	<1>		
649	00016DD0	0000000000000FE0606-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 0feh, 006h, 006h, 006h, 006h, 000h, 000h, 000h, 000h, 000h
649	00016DD9	060600000000000	<1>		
650	00016DE0	00c0c0c2C6CC183060-	<1>	db	000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh, 000h, 000h
650	00016DE9	CE9B060c1F0000	<1>		
651	00016DF0	00c0c0c2C6CC183066-	<1>	db	000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h, 006h, 000h

657	00016E50	DD77DD77DD77DD77DD-	<1>	db	0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h, 0ddh, 077h
657	00016E59	77DD77DD77DD77	<1>		
658	00016E60	1818181818181818-	<1>	db	018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
658	00016E69	1818181818181818	<1>		
659	00016E70	18181818181818F818-	<1>	db	018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
659	00016E79	1818181818181818	<1>		
660	00016E80	1818181818F818F818-	<1>	db	018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
660	00016E89	1818181818181818	<1>		
661	00016E90	36363636363636F636-	<1>	db	036h, 036h, 036h, 036h, 036h, 036h, 036h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
661	00016E99	3636363636363636	<1>		
662	00016EA0	00000000000000FE36-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 0feh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
662	00016EA9	3636363636363636	<1>		
663	00016EB0	0000000000F818F818-	<1>	db	000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
663	00016EB9	1818181818181818	<1>		
664	00016EC0	3636363636F606F636-	<1>	db	036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
664	00016EC9	3636363636363636	<1>		
665	00016ED0	363636363636363636-	<1>	db	036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
665	00016ED9	3636363636363636	<1>		
666	00016EE0	0000000000FE06F636-	<1>	db	000h, 000h, 000h, 000h, 000h, 0feh, 006h, 0f6h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
666	00016EE9	3636363636363636	<1>		
667	00016EF0	3636363636F606FE00-	<1>	db	036h, 036h, 036h, 036h, 036h, 0f6h, 006h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
667	00016EF9	0000000000000000	<1>		
668	00016F00	36363636363636FE00-	<1>	db	036h, 036h, 036h, 036h, 036h, 036h, 036h, 0feh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
668	00016F09	0000000000000000	<1>		
669	00016F10	1818181818F818F800-	<1>	db	018h, 018h, 018h, 018h, 018h, 0f8h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
669	00016F19	0000000000000000	<1>		
670	00016F20	00000000000000F818-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 0f8h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
670	00016F29	1818181818181818	<1>		
671	00016F30	181818181818181F00-	<1>	db	018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
671	00016F39	0000000000000000	<1>		
672	00016F40	18181818181818FF00-	<1>	db	018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
672	00016F49	0000000000000000	<1>		
673	00016F50	00000000000000FF18-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
673	00016F59	1818181818181818	<1>		
674	00016F60	181818181818181F18-	<1>	db	018h, 018h, 018h, 018h, 018h, 018h, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
674	00016F69	1818181818181818	<1>		
675	00016F70	00000000000000FF00-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
675	00016F79	0000000000000000	<1>		
676	00016F80	18181818181818FF18-	<1>	db	018h, 018h, 018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
676	00016F89	1818181818181818	<1>		
677	00016F90	181818181818181F18-	<1>	db	018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
677	00016F99	1818181818181818	<1>		
678	00016FA0	363636363636363736-	<1>	db	036h, 036h, 036h, 036h, 036h, 036h, 036h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
678	00016FA9	3636363636363636	<1>		
679	00016FB0	363636363637303F00-	<1>	db	036h, 036h, 036h, 036h, 036h, 037h, 030h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
679	00016FB9	0000000000000000	<1>		
680	00016FC0	00000000003F303736-	<1>	db	000h, 000h, 000h, 000h, 000h, 03fh, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
680	00016FC9	3636363636363636	<1>		
681	00016FD0	3636363636F700FF00-	<1>	db	036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
681	00016FD9	0000000000000000	<1>		
682	00016FE0	0000000000FF00F736-	<1>	db	000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
682	00016FE9	3636363636363636	<1>		
683	00016FF0	363636363637303736-	<1>	db	036h, 036h, 036h, 036h, 036h, 037h, 030h, 037h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
683	00016FF9	3636363636363636	<1>		
684	00017000	0000000000FF00FF00-	<1>	db	000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
684	00017009	0000000000000000	<1>		
685	00017010	3636363636F700F736-	<1>	db	036h, 036h, 036h, 036h, 036h, 0f7h, 000h, 0f7h, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
685	00017019	3636363636363636	<1>		
686	00017020	1818181818FF00FF00-	<1>	db	018h, 018h, 018h, 018h, 018h, 0ffh, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
686	00017029	0000000000000000	<1>		
687	00017030	36363636363636FF00-	<1>	db	036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
687	00017039	0000000000000000	<1>		
688	00017040	0000000000FF00FF18-	<1>	db	000h, 000h, 000h, 000h, 000h, 0ffh, 000h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
688	00017049	1818181818181818	<1>		
689	00017050	00000000000000FF36-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
689	00017059	3636363636363636	<1>		
690	00017060	363636363636363F00-	<1>	db	036h, 036h, 036h, 036h, 036h, 036h, 036h, 03fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
690	00017069	0000000000000000	<1>		
691	00017070	18181818181F181F00-	<1>	db	018h, 018h, 018h, 018h, 018h, 01fh, 018h, 01fh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
691	00017079	0000000000000000	<1>		
692	00017080	000000000001F181F18-	<1>	db	000h, 000h, 000h, 000h, 000h, 01fh, 018h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
692	00017089	1818181818181818	<1>		
693	00017090	0000000000000003F36-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 03fh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
693	00017099	3636363636363636	<1>		
694	000170A0	36363636363636FF36-	<1>	db	036h, 036h, 036h, 036h, 036h, 036h, 036h, 0ffh, 036h, 036h, 036h, 036h, 036h, 036h, 036h, 036h
694	000170A9	3636363636363636	<1>		
695	000170B0	1818181818FF18FF18-	<1>	db	018h, 018h, 018h, 018h, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
695	000170B9	1818181818181818	<1>		
696	000170C0	18181818181818F800-	<1>	db	018h, 018h, 018h, 018h, 018h, 018h, 018h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
696	000170C9	0000000000000000	<1>		
697	000170D0	0000000000000001F18-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 01fh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
697	000170D9	1818181818181818	<1>		
698	000170E0	FFFFFFFFFFFFFFFFFFFF-	<1>	db	0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
698	000170E9	FFFFFFFFFFFFFFFF	<1>		
699	000170F0	00000000000000FFFF-	<1>	db	000h, 000h, 000h, 000h, 000h, 000h, 000h, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh
699	000170F9	FFFFFFFFFFFFFFFF	<1>		
700	00017100	F0F0F0F0F0F0F0F0F0-	<1>	db	0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h, 0f0h
700	00017109	F0F0F0F0F0F0F0F0	<1>		
701	00017110	0F0F0F0F0F0F0F0F0F-	<1>	db	00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh, 00fh
701	00017119	0F0F0F0F0F0F0F0F	<1>		
702	00017120	FFFFFFFFFFFFFFFF0000-	<1>	db	0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
702	00017129	0000000000000000	<1>		
703	00017130	000000000076DCD8D8-	<1>	db	000h, 000h, 000h, 000h, 000h, 076h, 0dch, 0d8h, 0d8h, 0d8h, 0dch, 076h, 000h, 000h, 000h, 000h
703	00017139	D8DC760000000000	<1>		
704	00017140	000078CCCCCD8CCC6-	<1>	db	000h, 000h, 078h, 0cch, 0cch, 0cch, 0d8h, 0cch, 0c6h, 0c6h, 0c6h, 0cch, 000h, 000h, 000h, 000h
704	00017149	C6C6CC0000000000	<1>		
705	00017150	0000FEC6C6C0C0C0C0-	<1>	db	000h, 000h, 0feh, 0c6h, 0c6h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 0c0h, 000h, 000h, 000h, 000h
705	00017159	C0C0C00000000000	<1>		
706	00017160	00000000FE6C6C6C6C-	<1>	db	000h, 000h, 000h, 000h, 0feh, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h
706	00017169	6C6C6C0000000000	<1>		
707	00017170	000000FEC660301830-	<1>	db	000h, 000h, 000h, 0feh, 0c6h, 060h, 030h, 018h, 030h, 060h, 0c6h, 0feh, 000h, 000h, 000h, 000h
707	00017179	60C6FE0000000000	<1>		
708	00017180	00000000007ED8D8D8-	<1>	db	000h, 000h, 000h, 000h, 000h, 07eh, 0d8h, 0d8h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h
708	00017189	D8D8700000000000	<1>		
709	00017190	000000006666666666-	<1>	db	000h, 000h, 000h, 000h, 000h, 066h, 066h, 066h, 066h, 066h, 07ch, 060h, 060h, 0c0h, 000h, 000h
709	00017199	7C6060C000000000	<1>		
710	000171A0	0000000076DC181818-	<1>	db	000h, 000h, 000h, 000h, 076h, 0dch, 018h, 018h, 018h, 018h, 018h, 018h, 000h, 000h, 000h, 000h
710	000171A9	1818180000000000	<1>		
711	000171B0	0000007E183C666666-	<1>	db	000h, 000h, 000h, 07eh, 018h, 03ch, 066h, 066h, 066h, 03ch, 018h, 07eh, 000h, 000h, 000h, 000h
711	000171B9	3C187E0000000000	<1>		
712	000171C0	000000386CC6C6FEC6-	<1>	db	000h, 000h, 000h, 038h, 06ch, 0c6h, 0c6h, 0feh, 0c6h, 0c6h, 06ch, 038h, 000h, 000h, 000h, 000h
712	000171C9	C66C38000000			

```

719 00017230 00000000FE0000FE00- <1> db 000h, 000h, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 0feh, 000h, 000h, 000h, 000h, 000h
719 00017239 00FE000000000000 <1>
720 00017240 0000000018187E1818- <1> db 000h, 000h, 000h, 000h, 018h, 018h, 07eh, 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h
720 00017249 0000FF0000000000 <1>
721 00017250 00000030180C060C18- <1> db 000h, 000h, 000h, 030h, 018h, 00ch, 006h, 00ch, 018h, 030h, 000h, 07eh, 000h, 000h, 000h, 000h
721 00017259 30007E0000000000 <1>
722 00017260 0000000C1830603018- <1> db 000h, 000h, 000h, 00ch, 018h, 030h, 060h, 030h, 018h, 00ch, 000h, 07eh, 000h, 000h, 000h, 000h
722 00017269 0C007E0000000000 <1>
723 00017270 00000E1B1818181818- <1> db 000h, 000h, 00eh, 01bh, 01bh, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h
723 00017279 1818181818181818 <1>
724 00017280 1818181818181818D8- <1> db 018h, 018h, 018h, 018h, 018h, 018h, 018h, 018h, 0d8h, 0d8h, 0d8h, 070h, 000h, 000h, 000h, 000h
724 00017289 D8D8700000000000 <1>
725 00017290 000000001818007E00- <1> db 000h, 000h, 000h, 000h, 018h, 018h, 000h, 07eh, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h
725 00017299 1818000000000000 <1>
726 000172A0 000000000076DC0076- <1> db 000h, 000h, 000h, 000h, 000h, 076h, 0dch, 000h, 076h, 0dch, 000h, 000h, 000h, 000h, 000h, 000h
726 000172A9 DC00000000000000 <1>
727 000172B0 00386C6C380000000000- <1> db 000h, 038h, 06ch, 06ch, 038h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
727 000172B9 0000000000000000 <1>
728 000172C0 00000000000000001818- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
728 000172C9 0000000000000000 <1>
729 000172D0 00000000000000000018- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
729 000172D9 0000000000000000 <1>
730 000172E0 000F0C0C0C0C0CEC6C- <1> db 000h, 00fh, 00ch, 00ch, 00ch, 00ch, 00ch, 0ech, 06ch, 06ch, 03ch, 01ch, 000h, 000h, 000h, 000h
730 000172E9 6C3C1C0000000000 <1>
731 000172F0 00D86C6C6C6C6C0000- <1> db 000h, 0d8h, 06ch, 06ch, 06ch, 06ch, 06ch, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
731 000172F9 0000000000000000 <1>
732 00017300 0070D83060C8F80000- <1> db 000h, 070h, 0d8h, 030h, 060h, 0c8h, 0f8h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
732 00017309 0000000000000000 <1>
733 00017310 000000007C7C7C7C7C- <1> db 000h, 000h, 000h, 000h, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 07ch, 000h, 000h, 000h, 000h, 000h
733 00017319 7C7C000000000000 <1>
734 00017320 00000000000000000000- <1> db 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
734 00017329 0000000000000000 <1>
735 <1>
736 <1> ; 01/01/2021 (TRDOS 386 v2.0.3)
737 <1>
738 <1> %if 0
739 <1>
740 <1> vgafont14alt:
741 <1> db 01dh, 000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 022h
742 <1> db 000h, 063h, 063h, 063h, 022h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 02bh
743 <1> db 000h, 000h, 018h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 000h, 000h, 000h, 02dh, 000h, 000h
744 <1> db 000h, 000h, 000h, 000h, 0ffh, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 000h, 04dh, 000h, 000h, 0c3h
745 <1> db 0e7h, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h, 000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh
746 <1> db 099h, 018h, 018h, 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h
747 <1> db 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h
748 <1> db 0dbh, 0dbh, 0ffh, 066h, 066h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h
749 <1> db 03ch, 066h, 0c3h, 0c3h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
750 <1> db 018h, 018h, 03ch, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h, 030h, 061h
751 <1> db 0c3h, 0ffh, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 000h, 0e6h, 0ffh, 0dbh, 0dbh, 0dbh
752 <1> db 0dbh, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
753 <1> db 000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h, 000h, 000h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h
754 <1> db 000h, 000h, 091h, 000h, 000h, 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h
755 <1> db 000h, 09bh, 000h, 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h
756 <1> db 09dh, 000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 0ffh, 018h, 000h, 000h, 09eh
757 <1> db 000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 0f3h, 000h, 000h, 000h, 0f1h, 000h
758 <1> db 000h, 018h, 018h, 018h, 0ffh, 018h, 018h, 018h, 000h, 0ffh, 000h, 000h, 000h, 0f6h, 000h, 000h
759 <1> db 018h, 018h, 000h, 000h, 0ffh, 000h, 000h, 018h, 000h, 000h, 000h, 000h, 000h, 000h, 000h
760 <1> vgafont16alt:
761 <1> db 01dh, 000h, 000h, 000h, 000h, 000h, 024h, 066h, 0ffh, 066h, 024h, 000h, 000h, 000h, 000h, 000h, 000h
762 <1> db 000h, 030h, 000h, 000h, 03ch, 066h, 0c3h, 0c3h, 0dbh, 0dbh, 0c3h, 0c3h, 066h, 03ch, 000h, 000h
763 <1> db 000h, 000h, 04dh, 000h, 000h, 0c3h, 0e7h, 0ffh, 0ffh, 0dbh, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 000h
764 <1> db 000h, 000h, 000h, 054h, 000h, 000h, 0ffh, 0dbh, 099h, 018h, 018h, 018h, 018h, 018h, 018h, 03ch
765 <1> db 000h, 000h, 000h, 000h, 056h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch
766 <1> db 018h, 000h, 000h, 000h, 000h, 057h, 000h, 000h, 0c3h, 0c3h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh
767 <1> db 066h, 066h, 000h, 000h, 000h, 000h, 058h, 000h, 000h, 0c3h, 0c3h, 066h, 03ch, 018h, 018h, 03ch
768 <1> db 066h, 0c3h, 0c3h, 000h, 000h, 000h, 000h, 059h, 000h, 000h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h
769 <1> db 018h, 018h, 018h, 03ch, 000h, 000h, 000h, 000h, 05ah, 000h, 000h, 0ffh, 0c3h, 086h, 00ch, 018h
770 <1> db 030h, 060h, 0c1h, 0c3h, 0ffh, 000h, 000h, 000h, 000h, 000h, 06dh, 000h, 000h, 000h, 000h, 0e6h
771 <1> db 0ffh, 0dbh, 0dbh, 0dbh, 0dbh, 0dbh, 000h, 000h, 000h, 076h, 000h, 000h, 000h, 000h, 000h
772 <1> db 0c3h, 0c3h, 0c3h, 0c3h, 066h, 03ch, 018h, 000h, 000h, 000h, 000h, 077h, 000h, 000h, 000h, 000h
773 <1> db 000h, 0c3h, 0c3h, 0c3h, 0dbh, 0dbh, 0ffh, 066h, 000h, 000h, 000h, 000h, 078h, 000h, 000h, 000h
774 <1> db 000h, 000h, 0c3h, 066h, 03ch, 018h, 03ch, 066h, 0c3h, 000h, 000h, 000h, 000h, 091h, 000h, 000h
775 <1> db 000h, 000h, 000h, 06eh, 03bh, 01bh, 07eh, 0d8h, 0dch, 077h, 000h, 000h, 000h, 09bh, 000h
776 <1> db 018h, 018h, 07eh, 0c3h, 0c0h, 0c0h, 0c0h, 0c3h, 07eh, 018h, 018h, 000h, 000h, 000h, 000h, 09dh
777 <1> db 000h, 000h, 0c3h, 066h, 03ch, 018h, 0ffh, 018h, 0ffh, 018h, 018h, 000h, 000h, 000h, 000h, 000h
778 <1> db 09eh, 000h, 0fch, 066h, 066h, 07ch, 062h, 066h, 06fh, 066h, 066h, 0f3h, 000h, 000h, 000h, 000h
779 <1> db 000h, 0abh, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 060h, 0ceh, 09bh, 006h, 00ch, 01fh
780 <1> db 000h, 000h, 0ach, 000h, 0c0h, 0c0h, 0c2h, 0c6h, 0cch, 018h, 030h, 066h, 0ceh, 096h, 03eh, 006h
781 <1> db 006h, 000h, 000h, 000h
782 <1>
783 <1> %endif
3791
3792 ; 20/11/2020
3793 vbe2_bochs_vbios:
3794 ; db "BOCHS/QEMU"
3795 00017330 424F4348532F51454D- ; 26/11/2020
3795 00017339 552F5649525455414C-
3795 00017342 424F58
3796 ;vbe_vnumber equ vbe2_bochs_vbios + 28 ; "3" or "2"
3797 ; 26/11/2020
3798 vbe_vnumber equ vbe2_bochs_vbios + 30 ; "3" or "2"
3799 ; 15/11/2020
3800 vesa_vbe3_bios_msg:
3801 ;db " VESA VBE version 3 Video BIOS ..."
3802 00017345 205645534120564245- db " VESA VBE3 Video BIOS ..." ; 26/11/2020
3802 0001734E 3320566964656F2042-
3802 00017357 494F53202E2E2E
3803 0001735E 0D0A0D0A00 db 0Dh, 0Ah, 0Dh, 0Ah, 0
3804
3805 ; 28/02/2021
3806 00017363 18 truecolor: db 24
3807
3808 align 2
3809
3810 ; EPOCH Variables
3811 ; 13/04/2015 - Retro UNIX 386 v1 Beginning
3812 ; 09/04/2013 epoch variables
3813 ; Retro UNIX 8086 v1 Prototype: UNIXCOPY.ASM, 10/03/2013
3814 ;
3815 00017364 8207 year: dw 1970
3816 00017366 0100 month: dw 1
3817 00017368 0100 day: dw 1
3818 0001736A 0000 hour: dw 0
3819 0001736C 0000 minute: dw 0
3820 0001736E 0000 second: dw 0
3821
3822 DMonth:
3823 00017370 0000 dw 0
3824 00017372 1F00 dw 31
3825 00017374 3B00 dw 59
3826 00017376 5A00 dw 90
3827 00017378 7800 dw 120
3828 0001737A 9700 dw 151
3829 0001737C B500 dw 181

```

```

3830 0001737E D400          dw 212
3831 00017380 F300          dw 243
3832 00017382 1101          dw 273
3833 00017384 3001          dw 304
3834 00017386 4E01          dw 334
3835
3836          ; 15/11/2020
3837 00017388 00          db 0
3838          kernel_version_msg: ; 17/04/2021
3839          ;;;;db "TRDOS (386) Kernel v2.0.4 by Erdogan Tan"
3840          ;;;;db "TRDOS (386) Kernel v2.0.5 by Erdogan Tan" ; 11/08/2022
3841          ;;;;db "TRDOS (386) Kernel v2.0.6 by Erdogan Tan" ; 29/08/2023
3842          ;;;;db "TRDOS (386) Kernel v2.0.7 by Erdogan Tan" ; 20/10/2023
3843          ;db "TRDOS (386) Kernel v2.0.8 by Erdogan Tan" ; 16/05/2024
3844          ;db "TRDOS (386) Kernel v2.0.9 by Erdogan Tan" ; 20/08/2024
3845 00017389 5452444F5320283338- ;db "TRDOS (386) Kernel v2.0.10 by Erdogan Tan" ; 11/01/2025
3845 00017392 3629204B65726E656C-
3845 0001739B 2076322E302E313020-
3845 000173A4 6279204572646F6761-
3845 000173AD 6E2054616E
3846 000173B2 00
3847
3848          db 0
3849          ; 20/02/2017
3850          KERNELFSIZE equ $ ; 04/07/2016
3851
3852          bss_start:
3853          ABSOLUTE bss_start
3854
3855 000173B3 ?????????? alignb 8 ; 25/12/2016
3856
3857          ; 15/04/2016
3858          ; TRDOS 386 (TRDOS v2.0)
3859          ; 80 interrupts
3860          ; 11/03/2015
3861          ; Interrupt Descriptor Table (20/08/2014)
3862          idt:
3863          ;resb 64*8 ; INT 0 to INT 3Fh
3864          ; 15/04/2016
3865 000173B8 <res 280h> resb 80*8 ; INT 0 to INT 4Fh
3866
3867          idt_end:
3868
3869          ;alignb 4
3870
3871          task_state_segment:
3872          ; 24/03/2015
3873 00017638 ????          tss.link: resw 1
3874 0001763A ????          resw 1
3875          ; tss offset 4
3876 0001763C ?????????? tss.esp0: resd 1
3877 00017640 ????          tss.ss0: resw 1
3878 00017642 ????          resw 1
3879 00017644 ?????????? tss.esp1: resd 1
3880 00017648 ????          tss.ss1: resw 1
3881 0001764A ????          resw 1
3882 0001764C ?????????? tss.esp2: resd 1
3883 00017650 ????          tss.ss2: resw 1
3884 00017652 ????          resw 1
3885          ; tss offset 28
3886 00017654 ?????????? tss.CR3: resd 1
3887 00017658 ?????????? tss.eip: resd 1
3888 0001765C ?????????? tss.eflags: resd 1
3889          ; tss offset 40
3890 00017660 ?????????? tss.eax: resd 1
3891 00017664 ?????????? tss.ecx: resd 1
3892 00017668 ?????????? tss.edx: resd 1
3893 0001766C ?????????? tss.ebx: resd 1
3894 00017670 ?????????? tss.esp: resd 1
3895 00017674 ?????????? tss.ebp: resd 1
3896 00017678 ?????????? tss.esi: resd 1
3897 0001767C ?????????? tss.edi: resd 1
3898          ; tss offset 72
3899 00017680 ????          tss.ES: resw 1
3900 00017682 ????          resw 1
3901 00017684 ????          tss.CS: resw 1
3902 00017686 ????          resw 1
3903 00017688 ????          tss.SS: resw 1
3904 0001768A ????          resw 1
3905 0001768C ????          tss.DS: resw 1
3906 0001768E ????          resw 1
3907 00017690 ????          tss.FS: resw 1
3908 00017692 ????          resw 1
3909 00017694 ????          tss.GS: resw 1
3910 00017696 ????          resw 1
3911 00017698 ????          tss.LDTR: resw 1
3912 0001769A ????          resw 1
3913          ; tss offset 100
3914 0001769C ????          resw 1
3915 0001769E ????          tss.IOPB: resw 1
3916          ; tss offset 104
3917          tss_end:
3918
3919 000176A0 ?????????? k_page_dir: resd 1 ; Kernel's (System) Page Directory address
3920          ; (Physical address = Virtual address)
3921 000176A4 ?????????? memory_size: resd 1 ; memory size in pages
3922 000176A8 ?????????? free_pages: resd 1 ; number of free pages
3923 000176AC ?????????? next_page: resd 1 ; offset value in M.A.T. for
3924          ; first free page search
3925 000176B0 ?????????? last_page: resd 1 ; offset value in M.A.T. which
3926          ; next free page search will be
3927          ; stopped after it. (end of M.A.T.)
3928 000176B4 ?????????? first_page: resd 1 ; offset value in M.A.T. which
3929          ; first free page search
3930          ; will be started on it. (for user)
3931 000176B8 ?????????? mat_size: resd 1 ; Memory Allocation Table size in pages
3932
3933          ; 20/11/2020
3934          ;vbe2bios: resw 1 ; VBE2 video bios ID (bochs/qemu)
3935          ; ; (0B0C4h or 0B0C5h for bochs/plex86 vgabios)
3936
3937          ; 02/09/2014 (Retro UNIX 386 v1)
3938          ; 04/12/2013 (Retro UNIX 8086 v1)
3939 000176BC ????          CRT_START: resw 1 ; starting address in regen buffer
3940          ; NOTE: active page only
3941 000176BE <res 10h> CURSOR_POSN: resw 8 ; cursor positions for video pages
3942          ACTIVE_PAGE:
3943          pty: resb 1 ; current tty
3944          ; 01/07/2015 - 29/01/2016
3945 000176CF ??          ccolor: resb 1 ; current color attribute
3946          ; 26/10/2015
3947          ; 07/09/2014
3948 000176D0 <res 14h> ttychr: resw ntty+2 ; Character buffer (multiscreen)
3949

```

```

3950 ; 18/05/2015 (03/06/2013 - Retro UNIX 8086 v1 feature only!)
3951 000176E4 ???????? p_time:      resd 1      ; present time (for systime & sysmdate)
3952
3953 ; 18/05/2015 (16/08/2013 - Retro UNIX 8086 v1 feature only !)
3954 ; (open mode locks for pseudo TTYS)
3955 ; [ major tty locks (return error in any conflicts) ]
3956 000176E8 <res 14h> ttyl:      resw ntty+2 ; opening locks for TTYS.
3957
3958 ; 15/04/2015 (Retro UNIX 386 v1)
3959 ; 22/09/2013 (Retro UNIX 8086 v1)
3960 000176FC <res Ah> wlist:      resb ntty+2 ; wait channel list (0 to 9 for TTYS)
3961 ; 15/04/2015 (Retro UNIX 386 v1)
3962 ; 12/07/2014 -> sp_init set comm. parameters as 0E3h
3963 ; 0 means serial port is not available
3964 ;comprm: ; 25/06/2014
3965 00017706 ?? com1p:      resb 1      ;;0E3h
3966 00017707 ?? com2p:      resb 1      ;;0E3h
3967
3968 ; 17/11/2015
3969 ; request for response (from the terminal)
3970 00017708 ???? req_resp:   resw 1
3971 ; 07/11/2015
3972 0001770A ?? ccomport:  resb 1 ; current COM (serial) port
3973 ; (0= COM1, 1= COM2)
3974
3975 0001770B ?? comqr:      resb 1 ; 'query or response' sign (u9.s, 'sndc')
3976 ; 07/11/2015
3977 0001770C ???? rchar:      resw 1 ; last received char for COM 1 and COM 2
3978 0001770E ???? schar:      resw 1 ; last sent char for COM 1 and COM 2
3979
3980 ; 22/08/2014 (RTC)
3981 ; (Packed BCD)
3982 00017710 ?? time_seconds: resb 1
3983 00017711 ?? time_minutes: resb 1
3984 00017712 ?? time_hours:   resb 1
3985 00017713 ?? date_wday:    resb 1
3986 00017714 ?? date_day:     resb 1
3987 00017715 ?? date_month:  resb 1
3988 00017716 ?? date_year:   resb 1
3989 00017717 ?? date_century: resb 1
3990
3991 ; 24/01/2016
3992 00017718 ???????? RTC_LH:      resd 1
3993 0001771C ?? RTC_WAIT_FLAG: resb 1
3994 0001771D ?? USER_FLAG:    resb 1
3995 ; 19/05/2016
3996 ;RTC_second:
3997 0001771E ?? RTC_2Hz:      resb 1 ; from 2Hz interrupt to 1Hz timer event function
3998
3999 %include 'diskbss.s' ; UNINITIALIZED DISK (BIOS) DATA
1 <1> ; *****
2 <1> ; TRDOS386.ASM (TRDOS 386 Kernel) - v2.0.5 - diskbss.s
3 <1> ;
4 <1> ; Last Update: 06/08/2022 (Previous: 24/01/2016)
5 <1> ;
6 <1> ; Beginning: 24/01/2016
7 <1> ;
8 <1> ; Assembler: NASM version 2.15 (trdos386.s)
9 <1> ;
10 <1> ; Turkish Rational DOS
11 <1> ; Operating System Project v2.0 by ERDOGAN TAN (Beginning: 04/01/2016)
12 <1> ;
13 <1> ; Derived from 'Retro UNIX 386 kernel - v0.2.1.0' source code by Erdogan Tan
14 <1> ; 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> ; (Unitialized Disk Parameters Data section for 'DISKIO.INC')
22 <1> ;
23 0001771F ?? <1> alignb 2
24 <1> ;
25 <1> ; -----
26 <1> ; TIMER DATA AREA :
27 <1> ; -----
28 <1> ;
29 <1> TIMER_LH: ; 16/02/2015
30 <1> TIMER_LOW: resw 1 ; LOW WORD OF TIMER COUNT
31 <1> TIMER_HIGH: resw 1 ; HIGH WORD OF TIMER COUNT
32 <1> TIMER_OFL: resb 1 ; TIMER HAS ROLLED OVER SINCE LAST READ
33 <1> ;
34 <1> ; -----
35 <1> ; DISKETTE DATA AREAS :
36 <1> ; -----
37 <1> ;
38 <1> SEEK_STATUS: resb 1
39 <1> MOTOR_STATUS: resb 1
40 <1> MOTOR_COUNT: resb 1
41 <1> DSKETTE_STATUS: resb 1
42 <1> NEC_STATUS: resb 7
43 <1> ;
44 <1> ; -----
45 <1> ; ADDITIONAL MEDIA DATA :
46 <1> ; -----
47 <1> ;
48 <1> LASTRATE: resb 1
49 <1> HF_STATUS: resb 1
50 <1> ; 06/08/2022
51 <1> ;HF_ERROR: resb 1
52 <1> HF_INT_FLAG: resb 1
53 <1> ; 06/08/2022
54 <1> ;HF_CNTRL: resb 1
55 <1> ;DSK_STATE: resb 4
56 <1> ; 06/08/2022
57 <1> DSK_STATE: resb 2
58 <1> DSK_TRK: resb 2
59 <1> ;
60 <1> ; -----
61 <1> ; FIXED DISK DATA AREAS :
62 <1> ; -----
63 <1> ;
64 <1> DISK_STATUS1: resb 1 ; FIXED DISK STATUS
65 <1> HF_NUM: resb 1 ; COUNT OF FIXED DISK DRIVES
66 <1> CONTROL_BYTE: resb 1 ; HEAD CONTROL BYTE
67 <1> ;@PORT_OFF resb 1 ; RESERVED (PORT OFFSET)
68 <1> ;port1_off resb 1 ; Hard disk controller 1 - port offset
69 <1> ;port2_off resb 1 ; Hard disk controller 2 - port offset
70 <1> ;
71 0001773A ???? <1> alignb 4
72 <1> ;
73 <1> ;HF_TBL_VEC: resd 1 ; Primary master disk param. tbl. pointer
74 <1> ;HF1_TBL_VEC: resd 1 ; Primary slave disk param. tbl. pointer

```

```

75      <1> HF_TBL_VEC: ; 22/12/2014
76 0001773C ???????? <1> HDPM_TBL_VEC:      resd 1      ; Primary master disk param. tbl. pointer
77 00017740 ???????? <1> HDPS_TBL_VEC:      resd 1      ; Primary slave disk param. tbl. pointer
78 00017744 ???????? <1> HDSS_TBL_VEC:      resd 1      ; Secondary master disk param. tbl. pointer
79 00017748 ???????? <1> HDSS_TBL_VEC:      resd 1      ; Secondary slave disk param. tbl. pointer
80      <1>
81      <1> ; 03/01/2015
82 0001774C ?? <1> LBAMode:          resb 1
83      <1>
84      <1> ; *****
4000      <1>
4001      <1> ;;; Real Mode Data (10/07/2015 - BSS)
4002      <1>
4003      <1> ;alignb 2
4004      <1>
4005      <1> ; 10/01/2016
4006      <1> %include 'trdoskx.s' ; UNINITIALIZED KERNEL (Logical Drive & FS) DATA
      <1> ; *****
      <1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.10) - UNINITIALIZED DATA : trdoskx.s
      <1> ; -----
      <1> ; Last Update: 19/12/2025 (Previous: 01/09/2024 - Kernel v2.0.9)
      <1> ; -----
      <1> ; Beginning: 04/01/2016
      <1> ; -----
      <1> ; Assembler: NASM version 2.11 (trdos386.s)
      <1> ; -----
      <1> ; Derived from TRDOS Operating System v1.0 (8086) source code by Erdogan Tan
      <1> ; TRDOS2.ASM (09/11/2011)
      <1> ; *****
      <1> ; DRV_INIT.ASM [26/09/2009] Last Update: 07/08/2011
      <1> ; MAINPROG.ASM [17/01/2004] Last Update: 09/11/2011
      <1> ; DIR.ASM [17/01/2004] Last Update: 09/10/2011
      <1> ; CMD_INTR.ASM [29/01/2005] Last update: 09/11/2011
      <1> ; DRV_FAT.ASM [07/07/2009] Last update: 21/08/2011
      <1>
19 0001774D ?????? <1> alignb 4
20      <1>
21      <1> ; MAINPROG.ASM
22 00017750 ???????? <1> MainProgCfg_FileSize: resd 1 ; 14/04/2016
23 00017754 ???????? <1> MainProgCfg_LineOffset: resd 1 ; 14/04/2016
24      <1>
25 00017758 ???????? <1> Current_vo1Serial: resd 1
26      <1>
27 0001775C ???????? <1> Current_Dir_FCluster: resd 1
28      <1>
29 00017760 ?? <1> Current_Dir_Level: resb 1
30 00017761 ?? <1> Current_FATType: resb 1
31 00017762 ?? <1> Current_Drv: resb 1
32 00017763 ?? <1> Current_Dir_Drv: resb 1 ; '?'
33 00017764 ?? <1> resb 1 ; ':'
34 00017765 ?? <1> Current_Dir_Root: resb 1 ; '/'
35 00017766 <res 5Ah> <1> Current_Directory: resb 90
36 000177C0 ?? <1> End_Of_Current_Dir_Str: resb 1
37 000177C1 ?? <1> Current_Dir_StrLen: resb 1
38      <1>
39 000177C2 ?? <1> CursorColumn: resb 1
40 000177C3 ?? <1> CmdArgStart: resb 1
41      <1>
42      <1> ; 03/02/2016
43 000177C4 <res 4Eh> <1> Remark: resb 78
44      <1>
45 00017812 <res 50h> <1> CommandBuffer: resb 80
46      <1>
47      <1> ;TextBuffer: ;resb 256
48      <1> ; 04/07/2025
49 00017862 <res 80h> <1> TextBuffer: resb 128
50 000178E2 <res 80h> <1> RunPathBuffer: resb 128
51      <1>
52      <1> MasterBootBuff:
53 00017962 <res 1BEh> <1> MasterBootCode: resb 1BEh
54 00017B20 <res 40h> <1> PartitionTable: resb 64
55 00017B60 ??? <1> MBIDCode: resw 1
56      <1>
57      <1> PTable_Buffer:
58 00017B62 <res 40h> <1> PTable_hd0: resb 64
59 00017BA2 <res 40h> <1> PTable_hd1: resb 64
60 00017BE2 <res 40h> <1> PTable_hd2: resb 64
61 00017C22 <res 40h> <1> PTable_hd3: resb 64
62      <1> ; 15/07/2020
63      <1> ;PTable_ep0: resb 64
64      <1> ;PTable_ep1: resb 64
65      <1> ;PTable_ep2: resb 64
66      <1> ;PTable_ep3: resb 64
67      <1>
68      <1> ; 22/05/2024
69 00017C62 ???????? <1> HD_LBA_yes: resd 1
70      <1> ; 13/08/2020
71 00017C66 ?? <1> scount: resb 1 ; 16/05/2016 (diskio.s, 'int33h:')
72      <1> ; 22/05/2024
73      <1> ;resb 1
74      <1> ;resb 1
75      <1> ;resb 1
76      <1> ;HD_LBA_yes: resb 1
77 00017C67 ?? <1> PP_Counter: resb 1
78 00017C68 ?? <1> EP_Counter: resb 1
79      <1> ; 13/08/2020
80 00017C69 ?? <1> LD_Counter: resb 1
81      <1>
82      <1> ; 30/08/2020
83 00017C6A ???????? <1> MBR_EP_StartSector: resd 1
84      <1> ; Extd partition start sector as in MBR
85 00017C6E ???????? <1> EP_StartSector: resd 1 ; next extd partition start sector
86      <1> ; 15/07/2020
87      <1> ;resd 1
88      <1> ;resd 1
89      <1>
90      <1> ; 20/07/2020
91 00017C72 <res 200h> <1> DOSBootSectorBuff: resb 512
92      <1> ; 15/07/2020
93      <1> ;DOSBootSectorBuff: resb 446 ; 1BEh
94      <1> ;MiniPartitionTable: resb 64 ; 40h
95      <1> ;MiniPartitionMagic: resw 1 ; 02h
96      <1>
97      <1> FAT_BuffDescriptor:
98 00017E72 ???????? <1> FAT_CurrentCluster: resd 1
99 00017E76 ?? <1> FAT_BuffvalidData: resb 1
100 00017E77 ?? <1> FAT_BuffDrvName: resb 1
101 00017E78 ??? <1> FAT_BuffOffset: resw 1
102 00017E7A ???????? <1> FAT_BuffSector: resd 1
103      <1>
104 00017E7E ???????? <1> FAT_ClusterCounter: resd 1
105 00017E82 ???????? <1> LastCluster: resd 1
106      <1>
107      <1> ; 16/05/2016

```

```

108 <1> ;; 18/03/2016 (TRDOS v2.0)
109 <1> ;ClusterBuffer_Valid: resb 1
110 <1>
111 <1> ; 29/07/2022
112 <1> ;resb 1
113 <1> ; 02/12/2023
114 00017E86 ?? <1> P_TIMER: resb 1 ; diskette change check (2 seconds)
115 <1>
116 <1> Dir_BuffDescriptor:
117 00017E87 ?? <1> DirBuff_DRV: resb 1
118 00017E88 ?? <1> DirBuff_FATType: resb 1
119 00017E89 ?? <1> DirBuff_validData: resb 1
120 00017E8A ???? <1> DirBuff_CurrentEntry: resw 1
121 00017E8C ???? <1> DirBuff_LastEntry: resw 1
122 00017E8E ???????? <1> DirBuff_Cluster: resd 1
123 00017E92 ???? <1> DirBuffer_Size: resw 1
124 <1> ;DirBuff_EntryCounter: resw 1
125 <1>
126 <1> ; 01/02/2016
127 <1> ; these are on (real mode) segment 8000h and later
128 <1> ; FAT_Buffer: resb 1536 ; 3 sectors
129 <1> ; Dir_Buffer: resb 512*32
130 <1> ; Logical_DOSDisks: resb 6656 ; 26 * 256 bytes
131 <1>
132 <1> ; 18/01/2016
133 <1>
134 00017E94 ???????? <1> FreeClusterCount: resd 1
135 <1>
136 00017E98 ???????? <1> VolSize_Unit1: resd 1
137 00017E9C ???????? <1> VolSize_Unit2: resd 1
138 <1>
139 00017EA0 ???????? <1> Vol_Tot_Sec_Str_Start: resd 1
140 00017EA4 <res Ah> <1> Vol_Tot_Sec_Str: resb 10
141 00017EAE ?? <1> Vol_Tot_Sec_Str_End: resb 1
142 00017EAF ?? <1> resb 1
143 00017EB0 ???????? <1> Vol_Free_Sectors_Str_Start: resd 1
144 00017EB4 <res Ah> <1> Vol_Free_Sectors_Str: resb 10
145 00017EBE ?? <1> Vol_Free_Sectors_Str_End: resb 1
146 <1>
147 <1> ; 10/02/2016
148 00017EBF ?? <1> RUN_CDRV: resb 1 ; CMD_INTR.ASM ; 09/11/2011
149 <1>
150 <1> ; 24/01/2016
151 00017EC0 <res 80h> <1> PATH_Array: resb 128 ; DIR.ASM ; 09/10/2011
152 <1> ; 06/02/2016
153 00017F40 ???????? <1> CCD_DriveDT: resd 1 ; DIR.ASM ; (word)
154 00017F44 ?? <1> CCD_Level: resb 1 ; DIR.ASM
155 00017F45 ?? <1> Last_Dir_Level: resb 1 ; DIR.ASM
156 <1> ;
157 00017F46 ???? <1> CDLF_AttributesMask: resw 1 ; DIR.ASM
158 00017F48 ???????? <1> CDLF_FNAddress: resd 1 ; DIR.ASM (word)
159 00017F4C ???? <1> CDLF_DEType: resw 1 ; DIR.ASM
160 <1> ;
161 00017F4E ?? <1> CD_COMMAND: resb 1 ; DIR.ASM
162 <1>
163 00017F4F ?? <1> alignb 4
164 <1>
165 <1> ; 29/01/2016
166 00017F50 ?? <1> Program_Exit: resb 1 ; CMD_INTR.ASM ; 09/11/2011
167 <1>
168 <1> ;alignb 4
169 <1> ; 23/02/2016
170 00017F51 ?? <1> disk_rw_op: resb 1 ; 0 = disk read, 1 = disk write
171 <1> ;disk_rw_spt: resb 1 ; sectors per track (<= 63) /// (<256)
172 <1> ; 31/01/2016
173 00017F52 ?? <1> retry_count: resb 1 ; DISK_IO.ASM ; 20/07/2011 (CHS_RetryCount)
174 00017F53 ?? <1> disk_rw_err: resb 1 ; DISK_IO.ASM ; (Disk_IO_err_code)
175 00017F54 ???????? <1> sector_count: resd 1 ; DISK_IO.ASM ; (Disk_RW_SectorCount)
176 <1>
177 <1> ; 06/02/2016 (long name)
178 00017F58 ???? <1> FDE_AttrMask: resw 1 ; DIR.ASM
179 00017F5A ???? <1> AmbiguousFileName: resw 1 ; DIR.ASM
180 00017F5C ?? <1> PreviousAttr: resb 1 ; DIR.ASM
181 <1> ;
182 00017F5D ?? <1> LongNameFound: resb 1 ; DIR.ASM
183 00017F5E ?? <1> LFN_EntryLength: resb 1 ; DIR.ASM
184 00017F5F ?? <1> LFN_CheckSum: resb 1 ; DIR.ASM
185 00017F60 <res 84h> <1> LongFileName: resb 132 ; DIR.ASM
186 <1>
187 <1> ;PATH_Array_Ptr: resw 1 ; DIR.ASM
188 00017FE4 ?? <1> PATH_CDLevel: resb 1 ; DIR.ASM
189 00017FE5 ?? <1> PATH_Level: resb 1 ; DIR.ASM
190 <1>
191 <1> ; 07/02/2016
192 00017FE6 <res Dh> <1> Dir_File_Name: resb 13 ; DIR.ASM ; 09/10/2011
193 <1>
194 <1> ; 10/02/2016
195 00017FF3 <res Dh> <1> Dir_Entry_Name: resb 13 ; DIR.ASM
196 <1>
197 <1> alignb 2
198 <1>
199 00018000 ???? <1> AttributesMask: resw 1 ; CMD_INTR.ASM ; 09/11/2011
200 <1>
201 <1> ; 10/02/2016 (128 bytes -> 126 bytes)
202 <1> ; 08/02/2016
203 <1> ;FFF Structure (128 bytes) ; DIR.ASM ; 09/10/2011
204 00018002 ?? <1> FindFile_Drv: resb 1
205 00018003 <res 41h> <1> FindFile_Directory: resb 65
206 00018044 <res Dh> <1> FindFile_Name: resb 13
207 <1> FindFile_LongNameEntryLength:
208 00018051 ?? <1> FindFile_LongNameYes: resb 1 ; sign for longname procedures
209 <1> ;Above 80 bytes form
210 <1> ;TR-DOS Source/Destination File FullName Format/Structure
211 00018052 ???? <1> FindFile_AttributesMask: resw 1
212 00018054 <res 20h> <1> FindFile_DirEntry: resb 32
213 00018074 ???????? <1> FindFile_DirFirstCluster: resd 1
214 00018078 ???????? <1> FindFile_DirCluster: resd 1
215 0001807C ???? <1> FindFile_DirEntryNumber: resw 1
216 0001807E ???? <1> FindFile_MatchCounter: resw 1
217 00018080 ???? <1> FindFile_Reserved: resw 1 ; 06/03/2016
218 <1>
219 00018082 ???????? <1> First_Path_Pos: resd 1 ; DIR.ASM ; 09/10/2011
220 00018086 ???????? <1> Last_Slash_Pos: resd 1 ; DIR.ASM
221 <1>
222 <1> ; 10/02/2016
223 0001808A ???? <1> File_Count: resw 1 ; DIR.ASM ; 09/10/2011
224 0001808C ???? <1> Dir_Count: resw 1
225 0001808E ???????? <1> Total_FSize: resd 1
226 00018092 ???????? <1> TFS_Dec_Begin: resd 1
227 00018096 <res Ah> <1> resb 10
228 000180A0 ?? <1> TFS_Dec_End: resb 1
229 <1>
230 000180A1 ?? <1> PrintDir_RowCounter: resb 1
231 <1>

```



```

232 000180A2 ???? <1> alignb 4
233 <1> ; 15/02/2015 ('show' command variables)
234 000180A4 ???????? <1> Show_FDT: resd 1
235 000180A8 ???????? <1> Show_LDDDT: resd 1
236 000180AC ???????? <1> Show_Cluster: resd 1
237 000180B0 ???????? <1> Show_FileSize: resd 1
238 000180B4 ???????? <1> Show_FilePointer: resd 1
239 000180B8 ???? <1> Show_ClusterPointer: resw 1
240 000180BA ???? <1> Show_ClusterSize: resw 1
241 000180BC ?? <1> Show_RowCount: resb 1
242 <1>
243 000180BD ?????? <1> alignb 4
244 <1> ; 21/02/2016
245 000180C0 ???????? <1> DelFile_FNPointer: resd 1 ; ; CMD_INTR.ASM (word) ; 09/11/2011
246 <1> ; 27/02/2016
247 <1> ; DIR.ASM (09/10/2011)
248 000180C4 ???????? <1> DelFile_FCluster: resd 1
249 000180C8 ???? <1> DelFile_EntryCounter: resw 1
250 000180CA ?? <1> DelFile_LNEL: resb 1
251 <1> ; 17/07/2025
252 <1> ; resb 1
253 <1> ; 22/02/2016
254 000180CB ?? <1> UPDLMDT_CDirLevel: resb 1
255 000180CC ???????? <1> UPDLMDT_CDirFCluster: resd 1
256 <1>
257 <1> ; DIR.ASM
258 000180D0 ???????? <1> mkdir_DirName_Offset: resd 1
259 000180D4 ???????? <1> mkdir_FFCluster: resd 1
260 000180D8 ???????? <1> mkdir_LastDirCluster: resd 1
261 000180DC ???????? <1> mkdir_FreeSectors: resd 1
262 000180E0 ???? <1> mkdir_attrb: resw 1
263 000180E2 ?? <1> mkdir_SecPerClust: resb 1
264 000180E3 ?? <1> mkdir_add_new_Cluster: resb 1
265 000180E4 <res Dh> <1> mkdir_Name: resb 13
266 000180F1 ???? <1> resw 1 ; 01/03/2016
267 <1> ; 27/02/2016
268 000180F3 ?? <1> Rmdir_MultiClusters: resb 1
269 000180F4 ???????? <1> Rmdir_DirEntryOffset: resd 1 ; 01/03/2016 (word -> dword)
270 000180F8 ???????? <1> Rmdir_ParentDirCluster: resd 1
271 000180FC ???????? <1> Rmdir_DirLastCluster: resd 1
272 00018100 ???????? <1> Rmdir_PreviousCluster: resd 1
273 <1>
274 <1> ; 17/07/2025 - TRDOS 386 v2.0.10
275 00018104 ???????? <1> rmdir_dir_fcluster: resd 1
276 00018108 ?? <1> rmdir_drv: resb 1
277 <1>
278 00018109 ?????? <1> alignb 4
279 <1> ; DRV_FAT.ASM ; 21/08/2011
280 0001810C ???????? <1> gffc_next_free_cluster: resd 1
281 00018110 ???????? <1> gffc_first_free_cluster: resd 1
282 00018114 ???????? <1> gffc_last_free_cluster: resd 1
283 <1>
284 <1> ; 29/04/2016
285 <1> Cluster_Index: ; resd 1
286 <1> ; 22/02/2016
287 00018118 ???????? <1> ClusterValue: resd 1
288 <1> ; 04/03/2016
289 0001811C ?? <1> Attributes: resb 1
290 <1> ; ; CFS_error: resb 1 ; ; 01/03/2016
291 0001811D ?? <1> resb 1
292 0001811E ?? <1> CFS_OPType: resb 1
293 0001811F ?? <1> CFS_Drv: resb 1
294 00018120 ???????? <1> CFS_CC: resd 1
295 00018124 ???????? <1> CFS_FAT32FSINFOSEC: resd 1
296 00018128 ???????? <1> CFS_FAT32FC: resd 1
297 <1>
298 <1> ; 27/02/2016
299 <1> ; alignb 4
300 0001812C ???????? <1> glc_prevcluster: resd 1 ; DRV_FAT.ASM (21/08/2011)
301 <1> ; 22/10/2016
302 00018130 ???????? <1> glc_index: resd 1 ; Last Cluster Index (22/10/2016)
303 <1>
304 <1> ; DIR.ASM
305 00018134 ???? <1> DLN_EntryNumber: resw 1
306 00018136 ?? <1> DLN_40h: resb 1
307 <1> ; 28/02/2016
308 00018137 ?? <1> TCC_FATerr: resb 1 ; DRV_FAT.ASM
309 <1>
310 <1> alignb 4
311 <1> ; DIR.ASM (09/10/2011)
312 00018138 ???? <1> LCDE_EntryIndex: resw 1 ; LCDE_EntryOffset
313 0001813A ???? <1> LCDE_ClusterSN: resw 1
314 0001813C ???????? <1> LCDE_Cluster: resd 1
315 00018140 ???????? <1> LCDE_ByteOffset: resd 1
316 <1>
317 <1> ; alignb 4
318 <1> ; 06/03/2016 (word -> dword)
319 <1> ; CMD_INTR.ASM (01/08/2010)
320 00018144 ???????? <1> SourceFilePath: resd 1
321 00018148 ???????? <1> DestinationFilePath: resd 1
322 <1>
323 <1> ; alignb 4
324 <1> ; 06/03/2016
325 <1> ; FILE.ASM (09/10/2011)
326 <1> ; Source File Structure (same with 'Find File' Structure)
327 0001814C ?? <1> SourceFile_Drv: resb 1
328 0001814D <res 41h> <1> SourceFile_Directory: resb 65
329 0001814E <res Dh> <1> SourceFile_Name: resb 13
330 <1> SourceFile_LongNameEntryLength:
331 0001819B ?? <1> SourceFile_LongNameYes: resb 1 ; Sign for longname procedures
332 <1> ; Above 80 bytes
333 <1> ; is TR-DOS Source File FullName Format/Structure
334 0001819C ???? <1> SourceFile_AttributesMask: resw 1
335 0001819E <res 20h> <1> SourceFile_DirEntry: resb 32
336 000181BE ???????? <1> SourceFile_DirFirstCluster: resd 1
337 000181C2 ???????? <1> SourceFile_DirCluster: resd 1
338 000181C6 ???? <1> SourceFile_DirEntryNumber: resw 1
339 000181C8 ???? <1> SourceFile_MatchCounter: resw 1
340 <1> ; 16/03/2016
341 000181CA ?? <1> SourceFile_SecPerClust: resb 1
342 000181CB ?? <1> SourceFile_Reserved: resb 1
343 <1> ; Above is 128 bytes
344 <1>
345 <1> ; Destination File Structure (same with 'Find File' Structure)
346 000181CC ?? <1> DestinationFile_Drv: resb 1
347 000181CD <res 41h> <1> DestinationFile_Directory: resb 65
348 0001820E <res Dh> <1> DestinationFile_Name: resb 13
349 <1> DestinationFile_LongNameEntryLength:
350 0001821B ?? <1> DestinationFile_LongNameYes: resb 1 ; Sign for longname procedures
351 <1> ; Above 80 bytes
352 <1> ; is TR-DOS Destination File FullName Format/Structure
353 0001821C ???? <1> DestinationFile_AttributesMask: resw 1
354 0001821E <res 20h> <1> DestinationFile_DirEntry: resb 32
355 0001823E ???????? <1> DestinationFile_DirFirstCluster: resd 1

```

```

356 00018242 ???????? <1> DestinationFile_DirCluster: resd 1
357 00018246 ???? <1> DestinationFile_DirEntryNumber: resw 1
358 00018248 ???? <1> DestinationFile_MatchCounter: resw 1
359 <1> ; 16/03/2016
360 0001824A ?? <1> DestinationFile_SecPerClust: resb 1
361 0001824B ?? <1> DestinationFile_Reserved: resb 1
362 <1> ; Above is 128 bytes
363 <1>
364 <1> ; 24/04/2016
365 0001824C ???? <1> resw 1
366 <1>
367 <1> ; 10/03/2016
368 <1> ; FILE.ASM
369 0001824E ?? <1> move_cmd_phase: resb 1
370 0001824F ?? <1> msftdf_sf_df_drv: resb 1
371 00018250 ????????? <1> msftdf_drv_offset: resd 1
372 <1>
373 <1> ; 11/03/2016
374 <1> ; DRV_FAT.ASM (21/08/2011)
375 00018254 ????????? <1> FAT_anc_LCluster: resd 1
376 00018258 ????????? <1> FAT_anc_FFCluster: resd 1
377 <1>
378 <1> ;alignb 4
379 <1>
380 <1> ; 14/03/2016
381 <1> ; TRDOS 386 = TRDOS v2.0 feature only !
382 <1> ; 'allocate_memory_block' in 'memory.s'
383 0001825C ????????? <1> mem_ipg_count: resd 1 ; page count (for contiguous allocation)
384 00018260 ????????? <1> mem_pg_count: resd 1 ; page count (for count down)
385 00018264 ????????? <1> mem_aperture: resd 1 ; contiguous free pages (current)
386 00018268 ????????? <1> mem_max_aperture: resd 1 ; maximum value of contiguous free pages
387 0001826C ????????? <1> mem_pg_pos: resd 1 ; mem. position (page #) of current aperture
388 00018270 ????????? <1> mem_max_pg_pos: resd 1 ; mem. position (page #) of max. aperture
389 <1>
390 <1> ; 15/03/2016
391 <1> ; FILE.ASM ('copy_source_file_to_destination_file')
392 00018274 ?? <1> copy_cmd_phase: resb 1
393 00018275 ?? <1> csftdf_rw_err: resb 1
394 00018276 ?? <1> DestinationFileFound: resb 1
395 00018277 ?? <1> csftdf_cdrv: resb 1
396 00018278 ????????? <1> csftdf_filesize: resd 1
397 <1> ; TRDOS386 (TRDOS v2.0)
398 <1> MainProgCfg_mem_addr: ; 19/12/2025
399 0001827C ????????? <1> csftdf_sf_mem_addr: resd 1
400 <1> MainProgCfg_mem_bsize: ; 19/12/2025
401 00018280 ????????? <1> csftdf_sf_mem_bsize: resd 1
402 <1> ;
403 <1> MainProgCfg_cluster: ; 19/12/2025
404 00018284 ????????? <1> csftdf_sf_cluster: resd 1 ; 16/03/2016
405 00018288 ????????? <1> csftdf_df_cluster: resd 1
406 <1> MainProgCfg_r_size: ; 19/12/2025
407 <1> ; 16/03/2016
408 0001828C ????????? <1> csftdf_r_size: resd 1
409 00018290 ????????? <1> csftdf_w_size: resd 1
410 <1> MainProgCfg_rbytes: ; 19/12/2025
411 00018294 ????????? <1> csftdf_sf_rbytes: resd 1
412 00018298 ????????? <1> csftdf_df_wbytes: resd 1
413 0001829C ?? <1> csftdf_percentage: resb 1
414 <1> ; 17/03/2016
415 0001829D ?? <1> csftdf_videopage: resb 1
416 0001829E ???? <1> csftdf_cursorpos: resw 1
417 000182A0 ????????? <1> csftdf_sf_drv_dt: resd 1
418 000182A4 ????????? <1> csftdf_df_drv_dt: resd 1
419 <1> ; 01/09/2024
420 <1> ; 29/08/2024
421 <1> ;csftdf_df_dclust: resd 1
422 <1> ;csftdf_df_dindex: resd 1
423 <1>
424 <1> ; 21/03/2016
425 <1> ; 20/03/2016
426 <1> ; FILE.ASM
427 000182A8 ????????? <1> createfile_Name_Offset: resd 1
428 000182AC ????????? <1> createfile_FreeSectors: resd 1
429 000182B0 ????????? <1> createfile_size: resd 1
430 000182B4 ????????? <1> createfile_FFCluster: resd 1 ; 11/03/2016
431 000182B8 ????????? <1> createfile_LastDirCluster: resd 1
432 000182BC ????????? <1> createfile_Cluster: resd 1
433 000182C0 ????????? <1> createfile_PCluster: resd 1
434 000182C4 ?? <1> createfile_attr: resb 1
435 000182C5 ?? <1> createfile_SecPerClust: resb 1
436 000182C6 ???? <1> createfile_DirIndex: resw 1
437 000182C8 ????????? <1> createfile_CCount: resd 1
438 <1> ; 19/12/2025
439 <1> ;createfile_BytesPerSec: resw 1 ; 23/03/2016
440 000182CC ?? <1> createfile_wfc: resb 1
441 000182CD ?? <1> createfile_UpdatePDir: resb 1 ; 31/03/2016
442 <1>
443 <1> ;alignb 4
444 <1>
445 <1> ; 11/04/2016
446 000182CE ???? <1> env_var_length: resw 1
447 <1>
448 <1> alignb 4
449 <1>
450 <1> ; 25/04/2016
451 000182D0 ?? <1> readi.valid: resb 1 ; valid data (>0 = valid for readi)
452 000182D1 ?? <1> readi.drv: resb 1 ; drive number (0, 1,2,3,4..)
453 000182D2 ?? <1> readi.spc: resb 1 ; sectors per cluster for 'readi' drive
454 000182D3 ?? <1> readi.s_index: resb 1 ; sector index in current cluster (buffer)
455 000182D4 ????????? <1> readi.sector: resd 1 ; current disk sector
456 000182D8 ???? <1> readi.bpc: resw 1 ; bytes per cluster - 1
457 000182DA ???? <1> readi.offset: resw 1 ; byte offset in cluster buffer
458 000182DC ????????? <1> readi.cluster: resd 1 ; current cluster number
459 000182E0 ????????? <1> readi.c_index: resd 1 ; cluster index of the current cluster (0,1,2,3..)
460 000182E4 ????????? <1> readi.fclust: resd 1 ; first cluster of the current cluster
461 <1> ; 19/12/2025
462 <1> ;readi.fs_index: resd 1 ; sector index in disk/file section (for Singlix FS)
463 <1> ;readi.buffer: resd 1 ; readi sector buffer address
464 <1>
465 <1> ;alignb 4
466 <1>
467 000182E8 ?? <1> writei.valid: resb 1 ; valid data (>0 = valid for writei)
468 000182E9 ?? <1> writei.drv: resb 1 ; drive number (0, 1,2,3,4..)
469 000182EA ?? <1> writei.spc: resb 1 ; sectors per cluster for 'writei' drive
470 000182EB ?? <1> writei.s_index: resb 1 ; sector index in current cluster (buffer)
471 000182EC ????????? <1> writei.sector: resd 1 ; current disk sector
472 000182F0 ???? <1> writei.bpc: resw 1 ; bytes per cluster - 1
473 000182F2 ???? <1> writei.offset: resw 1 ; byte offset in cluster buffer
474 000182F4 ????????? <1> writei.cluster: resd 1 ; current cluster number
475 000182F8 ????????? <1> writei.c_index: resd 1 ; cluster index of the current cluster (0,1,2,3..)
476 000182FC ????????? <1> writei.fclust: resd 1 ; first cluster of the current cluster
477 00018300 ????????? <1> writei.fs_index: resd 1 ; sector index in disk/file section (for Singlix FS)
478 <1> ;writei.buffer: resd 1 ; writei sector buffer address
479 00018304 ????????? <1> writei.lclust: resd 1 ; writei last cluster (mget_w) ; 23/10/2016

```

```

480 00018308 ???????? <1> writei.l_index: resd 1 ; writei last cluster index (mget_w) ; 23/10/2016
481 0001830C ?? <1> writei.ofn: resb 1 ; open file number (to be written) ; 23/10/2016
482 <1>
483 0001830D ?????? <1> alignb 4
484 <1>
485 <1> ; 29/04/2016
486 00018310 ???????? <1> Run_CDirFC: resd 1
487 <1> ; Run_Auto_Path: resb 1
488 <1> ; 03/07/2025
489 <1> Run_Path_Length:
490 00018314 ???????? <1> resd 1
491 00018318 ???????? <1> Run_Auto_Path: resd 1
492 <1> Run_Manual_Path:
493 0001831C ?? <1> resb 1 ; 0 -> auto path sequence needed
494 0001831D ?? <1> EXE_ID: resb 1
495 0001831E ?? <1> EXE_dot: resb 1
496 0001831F ?? <1> resb 1 ; 03/07/2025
497 <1>
498 <1> ; 06/05/2016
499 00018320 ???????? <1> mainprog_return_addr: resd 1
500 00018324 ???????? <1> last_error: resd 1 ; this will be used to return error code to MainProg
501 <1> ; 'lasterror' keyword will be used later to get the
502 <1> ; last error code/number/status.
503 <1> ; 12/05/2016
504 00018328 ???????? <1> video_eax: resd 1 ; eax return value of video function
505 <1>
506 <1> ; 01/06/2016
507 0001832C ???????? <1> user_buffer: resd 1 ; 'diskio.s' (INT 33h, Function 08h, floppy disk type)
508 <1>
509 <1> ; 21/05/2016 - TRDOS 386 ('swap/switch', 'rswap', [u.pri])
510 00018330 ?? <1> priority: resb 1 ; running priority level of process (0,1,2)
511 <1> ; (run queue which is process comes from)
512 <1> ; 22/05/2016 - TRDOS 386 ('set_run_sequence', 'rtc_int', 'u_timer')
513 00018331 ?? <1> p_change: resb 1 ; process change status (for timer events)
514 <1> ; 23/05/2016 - TRDOS 386 ('clock')
515 00018332 ?? <1> multi_tasking: resb 1 ; Multi Tasking status (0 = disabled, >0 = enabled)
516 <1> ; (EBX will return with user buffer addr or disk type)
517 <1> ; 07/06/2016
518 00018333 ?? <1> timer_events: resb 1 ; number of (active) timer events, <= 16
519 <1>
520 <1> ; 24/06/2016
521 00018334 ?? <1> w_str_cmd: resb 1 ; WRITE_STRING command (0,1,2,3) ; video.s
522 00018335 ?? <1> p_crt_mode: resb 1 ; previous video mode (=3 or 0), backup mark/sign
523 <1> ; 26/06/2016
524 00018336 ?? <1> p_crt_page: resb 1 ; previous active page (for 'set_mode')
525 <1> ; 04/07/2016
526 00018337 ?? <1> noclearmem: resb 1 ; if set, 'SET MODE' (INT 31h) function (AH = 4)
527 <1> ; will not clear the video memory
528 <1> ; (usable for graphics modes only)
529 <1> alignb 2
530 00018338 ???? <1> CRT_LEN: resw 1 ; length of regen buffer in bytes
531 0001833A <res 10h> <1> cursor_pposn: resw 8 ; cursor positions backup
532 <1>
533 <1> ; 10/07/2016 ('VGA_FONT_SETUP', INT 43H address for x86 real mode bios)
534 0001834A ???????? <1> VGA_INT43H: resd 1 ; 0 = default (not configured by user)
535 <1> ; 0FFFFFFFh = user defined fonts
536 <1> ; address:
537 <1> ; vgafont8
538 <1> ; vgafont16
539 <1> ; vgafont14
540 <1>
541 <1> ; 25/07/2016
542 0001834E ?? <1> VGA_MTYPE: resb 1 ; 0=CTEXT,1=MTEXT,2=CGA,3=PLANAR1,4=PLANAR4,5=LINEAR
543 <1>
544 <1> ; 23/10/2016
545 0001834F ?? <1> setfmod: resb 1 ; update last modification date&time sign (if >0)
546 <1> ; (it is Open File Number + 1, if > 0)
547 <1> ; 27/08/2024
548 00018350 ?? <1> setfclust: resb 1 ; first cluster of file
549 <1> ; (is used by update lmdt proc)
550 00018351 ?????? <1> alignb 4
551 <1>
552 <1> ; 16/10/2016
553 00018354 ???????? <1> FFF_UBuffer: resd 1 ; User's buffer address for FFF & FNF system calls
554 <1> ; 15/10/2016
555 00018358 ?? <1> FFF_Valid: resb 1 ; Find First File Structure validation byte
556 <1> ; 0 = invalid (Find Next File can't use FFF struct)
557 <1> ; >0 = valid, return type for FFF and Find Next File
558 <1> ; 24 = basic parameters, 24 bytes
559 <1> ; 128 = entire FFF structure/table, 128 bytes
560 <1> ; 16/10/2016 (FFF_Attrib: resw 1)
561 00018359 ?? <1> FFF_Attrib: resb 1 ; Find First File attributes for Find Next File (LB)
562 0001835A ?? <1> FFF_RType: resb 1 ; FFF return type (0 = Basic, >0 = complete) (HB)
563 <1> ; 16/10/2016 - 05/10/2016 (Set Working Path)
564 0001835B ?? <1> SWP_inv_fname: resb 1 ; Set Working Path - Invalid File Name
565 0001835C ???? <1> SWP_Mode: resw 1 ; Set Working Path - Mode
566 0001835E ?? <1> SWP_DRV: resb 1 ; Set Working Path - Drive
567 0001835F ?? <1> SWP_DRV_chg: resb 1 ; Set Working Path - Drive Change
568 <1>
569 <1> ; 27/02/2017
570 00018360 ?? <1> fpready: resb 1 ; '80387 fpu is ready' flag
571 <1>
572 <1> ; 17/04/2021
573 <1> ; (DEVICE parameters is disabled as temporary)
574 <1>
575 <1> ; 08/10/2016
576 <1> ;device_name: resb 9 ; capitalized (and zero padded) device name
577 <1> ; (example: "TTY0",0,0,0,0,0)
578 <1>
579 00018361 ?????? <1> alignb 4
580 <1>
581 <1> ; 08/10/2016
582 <1> ; 07/10/2016
583 <1> ; Table of kernel devices (which do not use installable device drivers)
584 <1> ; has been coded into KERNEL (trdosk9.s)
585 <1> ; 07/10/2016
586 <1> ; 8 installable device drivers available to install (NUMIDEV)
587 <1> ;IDEV_PGDIR: resd NUMIDEV
588 <1> ; Page directories of installable device drivers
589 <1>
590 <1> ; Note: virtual start address is always 400000h
591 <1> ; (end of the 1st 4MB). [org 400000h]
592 <1> ; Segments: KCODE, KDATA
593 <1> ; Method: call 400000h (after changing page dir)
594 <1> ; Query code located at the start (400000h).
595 <1> ; Query code returns with
596 <1> ; eax = device type and driver version
597 <1> ; AL = Device Type minor
598 <1> ; AH = Device Type major
599 <1> ; Byte 16-23 : Version minor
600 <1> ; Byte 24-31 : Version major - 1
601 <1> ; (0:0 -> 1.0)
602 <1> ; ebx = initialization code address
603 <1> ; ecx = configuration table address

```

```

604      <1>      ;     edx = description table address
605      <1>      ;     esi = device (default) name address (ASCIIZ)
606      <1>      ;     (name has "/DEV/" prefix)
607      <1>      ;     edi = dispatch table address
608      <1>      ;     (for calling kernel-device functions)
609      <1>      ;     ebp = address table address
610      <1>      ; Initialization code returns with
611      <1>      ;     eax = open code address
612      <1>      ;     ecx = close code address
613      <1>      ;     ebx = read code address
614      <1>      ;     edx = write code address
615      <1>      ;     esi = IOCTL code address
616      <1>      ;     edi = dispatch table address
617      <1>      ;     ebp = address table address
618      <1>      ; Address Table:
619      <1>      ;     Offset 0 : open code address
620      <1>      ;     Offset 4 : read code address
621      <1>      ;     Offset 8 : write code address
622      <1>      ;     Offset 12 : close code address
623      <1>      ;     Offset 16 : IOCTL code address
624      <1>      ;     Offset 20 : initialization code address
625      <1>      ;     Offset 24 : description table address
626      <1>      ;     Offset 28 : configuration table address
627      <1>      ;     Offset 32 : device name address
628      <1>      ;     Offset 36 : dispatch table address
629      <1>      ;     (for calling kernel-device functions)
630      <1>
631      <1> ;IDEV_NAME:  resb 8*NUMIDEV
632      <1>      ; 8 byte names of installable device drivers
633      <1>
634      <1> ;IDEV_TYPE:  resb NUMIDEV ; Driver type of installable device drivers
635      <1> ;IDEV_FLAGS: resb NUMIDEV ; Device access parameters for installable
636      <1>      ; device drivers (These values are set while
637      <1>      ; the device driver is being loaded.)
638      <1> ;IDEV_OADDR: resd NUMIDEV ; open function addr for installable dev driver
639      <1> ;IDEV_CADDR: resd NUMIDEV ; close function addr for installable dev driver
640      <1> ;IDEV_RADDR: resd NUMIDEV ; read function addr for installable dev driver
641      <1> ;IDEV_WADDR: resd NUMIDEV ; write function addr for installable dev driver
642      <1>
643      <1> ; 08/10/2016
644      <1> ; 07/10/2016
645      <1> ; Device Open and Access parameters
646      <1> ;DEV_ACCESS:  resb NUMOFDEVICES ; bit 0 = accessible by normal users
647      <1>      ; bit 1 = read access permission
648      <1>      ; bit 2 = write access permission
649      <1>      ; bit 3 = IOCTL permission to users
650      <1>      ; bit 4 = block device if it is set
651      <1>      ; bit 5 = 16 bit or 1024 byte data
652      <1>      ; bit 6 = 32 bit or 2048 byte data
653      <1>      ; bit 7 = installable device driver
654      <1> ;DEV_R_OWNER:  resb NUMOFDEVICES ; Reading owner no (u.uid) of devices
655      <1> ;DEV_R_OPENCOUNT: resb NUMOFDEVICES ; Reading open count
656      <1> ;DEV_W_OWNER:  resb NUMOFDEVICES ; Writing owner no (u.uid) of devices
657      <1> ;DEV_W_OPENCOUNT: resb NUMOFDEVICES ; Writing open count
658      <1> ;DEV_DRIVER:  resb NUMOFDEVICES ; device driver number (1 to 7Fh)
659      <1>      ; *if bit 7 is set (80 to FFh)
660      <1>      ; *if it is installable device driver
661      <1>      ; *index (0 to 7Fh)
662      <1>      ; otherwise it is kernel device index
663      <1> ;DEV_OPENMODE: resb NUMOFDEVICES ; 1 = read mode
664      <1>      ; 2 = write mode
665      <1>      ; 3 = read & write
666      <1>      ; 0 = not open (free)
667      <1> ;DEV_NAME_PTR:  resd NUMOFDEVICES ; pointers to name addresses of drivers
668      <1>      ; Address base: KDEV_NAME+
669      <1>      ; or IDEV_NAME+
670      <1> ;DEV_R_POINTER:  resd NUMOFDEVICES ; reading pointer, writing pointer
671      <1> ;DEV_W_POINTER:  resd NUMOFDEVICES ; sector number if block device
672      <1>      ; character offset if char device
673      <1> alignb 4
674      <1>
675      <1> ; 06/10/2016
676      <1> ; Open File Parameters
677      <1> OF_FCLUSTER:  resd OPENFILES ; First clusters of open files
678      <1> OF_DRIVE:  resb OPENFILES ; Logical DOS drive numbers of open files
679      <1> OF_MODE:  resb OPENFILES ; Open mode (1 = read, 2 = write, 3 = r&w)
680      <1> OF_STATUS: resb OPENFILES ; (bit 0 = read, bit 1 = write)
681      <1> OF_OPENCOUNT: resb OPENFILES ; Open counts of open files
682      <1> OF_POINTER: resd OPENFILES ; File seek/read/write pointer
683      <1> OF_SIZE:  resd OPENFILES ; File sizes of open files (in bytes)
684      <1> OF_DIRFCLUSTER: resd OPENFILES ; Directory First Clusters of open files
685      <1> OF_DIRCLUSTER:  resd OPENFILES ; Directory (Entry) Clusters of open files
686      <1> OF_VOLUMEID:  resd OPENFILES ; Vol ID for removable drives of open files
687      <1> OF_CCLUSTER:  resd OPENFILES ; Current clusters of open files
688      <1> OF_CCINDEX:  resd OPENFILES ; Cluster index numbers of current clusters
689      <1> ; 24/10/2016
690      <1> OF_DIRENTRY:  resw OPENFILES ; Directory entry index no. in dir cluster
691      <1>      ; Sector index = entry index / 16
692      <1>
693      <1> ;alignb 2
694      <1>
695      <1> DTA:          ;resd 24      ; Find First File data transfer area
696      <1>      resb 24      ; 29/07/2022
697      <1>
698      <1> ; 19/12/2016
699      <1> tcallback: resb 1      ; Timer callback method flag for 'systimer'
700      <1> trtc:      resb 1      ; Timer interrupt type flag for 'systimer'
701      <1> ; 20/02/2017
702      <1> no_page_swap: resb 1      ; Swap lock for Signal Response Byte pages
703      <1> ;15/01/2017
704      <1> ; 02/01/2017
705      <1> ;;intflg:  resb 1      ; software interrupt in progress signal
706      <1>      ; (for timer interrupt)
707      <1> alignb 4
708      <1> ; 13/04/2017
709      <1> ;DEV_INTR:  resb NUMOFDEVICES ; Device Interrupt (IRQ) number + 1
710      <1>      ; (0= not available, 1= IRQ 0, 16= IRQ 15)
711      <1> DEV_INT_HNDLR:  resd 16      ; Device Interrupt Handler addr, if > 0
712      <1>
713      <1> ;alignb 4
714      <1>
715      <1> ; 26/02/2017 ; IRQ Callback parameters ('syscalbac')
716      <1> ;Index:  ; 0 to 8
717      <1> ; 0 = IRQ3, 1 = IRQ4, 2 = IRQ5, 3 = IRQ7
718      <1> ; 4 = IRQ9, 5 = IRQ10, 6 = IRQ11, 7 = IRQ12, 8 = IRQ13
719      <1> IRQ.owner:  resb 9      ; owner, 0 = free, >0 = [u.uno]
720      <1> IRQ.dev:  resb 9      ; 0 = default/kernel, >0 = device number
721      <1> IRQ.method: resb 9      ; 0 = Signal Response Byte, 1 = Callback
722      <1> IRQ.srb:  resb 9      ; Signal Response/Return Byte value
723      <1> IRQ.addr:  resd 9      ; Signal Response Byte address (physical)
724      <1>      ; or Callback service address (virtual)
725      <1> ; 28/02/2017
726      <1> IRQ_cr3:  resd 1      ; for saving cr3 register in IRQ handler
727      <1> IRQnum:  resb 1      ; IRQ number for IRQ handler (trdosk8.s)

```

```

728 <1> ; 10/04/2017
729 <1> ; 03/04/2017
730 <1> ; UNINITIALIZED AUDIO DATA
731 000188CD ?????? <1> alignb 4
732 000188D0 ?? <1> audio_pci: resb 1
733 000188D1 ?? <1> audio_device: resb 1
734 <1> ;audio_mode: resb 1
735 000188D2 ?? <1> audio_intr: resb 1
736 000188D3 ?? <1> audio_busy: resb 1 ; Busy flag for audio irq ; 21/04/2017
737 <1> ;audio_reserved: resb 1
738 <1> ; 20/11/2023
739 000188D4 ???? <1> NAMBAR: resw 1 ; Native Audio Mixer Base Address
740 <1> NABMBAR: ; 02/10/2023 (NABMBAR = audio_io_base)
741 000188D6 ???? <1> audio_io_base: resw 1 ; Base I/O address of audio device
742 000188D8 ???????? <1> audio_dev_id: resd 1 ; BUS/DEV/FN ; 00000000BBBBBBBBDDDDFF00000000
743 000188DC ???????? <1> audio_vendor: resd 1
744 000188E0 ???????? <1> audio_stats_cmd: resd 1
745 <1> ;
746 000188E4 ???????? <1> audio_buffer: resd 1 ; virtual address of user's audio buffer
747 000188E8 ???????? <1> audio_p_buffer: resd 1 ; Physical address of user's audio buffer
748 000188EC ???????? <1> audio_buff_size: resd 1 ; user's audio buffer size (half buffer size)
749 000188F0 ???????? <1> audio_dma_buff: resd 1 ; dma buffer address
750 000188F4 ???????? <1> audio_dmabuff_size: resd 1 ; dma buffer size (2 * half buffer size)
751 <1> ; 05/06/2024
752 000188F8 ???????? <1> dma_hbuff_size: resd 1 ; dma half buffer size
753 <1> ;
754 000188FC ?? <1> audio_flag: resb 1 ; dma buffer flag (1st half = 0, 2nd half = 1)
755 000188FD ?? <1> audio_user: resb 1 ; user number of the owner
756 000188FE ?? <1> audio_cb_mode: resb 1 ; 0 = signal response byte method
757 <1> ; 1 = callback method
758 <1> ; 2 = s.r.b. method with auto increment
759 000188FF ?? <1> audio_srb: resb 1 ; signal response byte value
760 00018900 ???????? <1> audio_cb_addr: resd 1 ; callback service address or s.r.b. address
761 <1> ; (s.r.b. addr is physical, cbs addr is virtual)
762 <1> ;
763 00018904 ?? <1> audio_bps: resb 1 ; selected mode: 8 bit, 16 bit
764 00018905 ?? <1> audio_stmo: resb 1 ; selected mode: mono /stereo
765 00018906 ???? <1> audio_freq: resw 1 ; sampling rate
766 <1> ; 20/11/2023
767 00018908 ?? <1> VRA: resb 1
768 00018909 ?? <1> audio_mode: resb 1
769 <1> ;
770 <1> ; 21/04/2017
771 0001890A ?? <1> audio_play_cmd: resb 1 ; Play/Stop command (1 = play, 0 = stop)
772 <1> ; 21/11/2023
773 <1> ;audio_civ: ; 28/05/2017 ; Current Buffer Index (AC'97)
774 <1> ; 23/05/2024
775 <1> LVI: ; AC'97 Last Valid Buffer Index
776 0001890B ?? <1> audio_flag_eol: resb 1 ; End of Link status (vt8233, EOL/FLAG)
777 <1> ;
778 <1> audio_master_volume:
779 0001890C ?? <1> audio_master_volume_l: resb 1 ; sound volume (lineout) left channel
780 0001890D ?? <1> audio_master_volume_r: resb 1 ; sound volume (lineout) right channel
781 <1> ;
782 <1> ; 24/05/2024
783 <1> audio_pcmo_volume:
784 0001890E ?? <1> audio_pcmo_volume_l: resb 1 ; PCM out volume left channel
785 0001890F ?? <1> audio_pcmo_volume_r: resb 1 ; PCM out volume right channel
786 <1> ;
787 <1> ; 02/06/2024
788 <1> ;alignb 4
789 <1> ;
790 <1> ; 20/11/2023
791 <1> ; 28/05/2017
792 <1> ; AC'97 Audio Controller Base Address Registers
793 <1> ;NAMBAR: resw 1 ; Native Audio Mixer Base Address
794 <1> ;NABMBAR: resw 1 ; Native Audio Bus Mastering Base Address
795 <1> ;
796 <1> ; 02/06/2024
797 <1> alignb 8
798 <1> ;
799 <1> ; 21/04/2017
800 00018910 <res 400h> <1> audio_bdl_buff: resd 32*8 ; VT8233 (AC97) BDL Buffer Size
801 <1> ; 12/05/2017
802 00018D10 ???????? <1> base_addr: resd 1 ; 'direct_memory_access' (memory.s)
803 <1> ;
804 <1> ; 01/06/2024
805 00018D14 ?? <1> reset: resb 1 ; AC97 init
806 <1> ;
807 <1> ; 28/08/2017
808 <1> ; 20/08/2017
809 <1> ;resb 1 ;
810 00018D15 ?? <1> dma_user: resb 1 ; user number for sysdma
811 00018D16 ?? <1> dma_channel: resb 1 ; dma channel for sysdma
812 00018D17 ?? <1> dma_mode: resb 1 ; dma mode for sysdma
813 00018D18 ???????? <1> dma_addr: resd 1 ; dma buffer physical addr for sysdma
814 00018D1C ???????? <1> dma_size: resd 1 ; dma buffer size (in bytes) for sysdma
815 00018D20 ???????? <1> dma_start: resd 1 ; dma start address for sysdma
816 00018D24 ???????? <1> dma_count: resd 1 ; dma count (in bytes) for sysdma
817 <1> ;
818 <1> ; 04/12/2023
819 <1> %if 0
820 <1> ;
821 <1> alignb 65536
822 <1> ; 09/08/2017
823 <1> ; 12/05/2017
824 <1> sb16_dma_buffer: resb 65536 ; DMA buffer for sb16 audio playing.
825 <1> ;
826 <1> %endif
4007 <1> ; 24/01/2016
4008 <1> %include 'ubss.s' ; UNINITIALIZED KERNEL (USER) DATA
<1> ; *****
<1> ; TRDOS386.ASM (TRDOS 386 Kernel - v2.0.9) - UNINITIALIZED USER DATA : ubss.s
<1> ; -----
<1> ; Last Update: 19/08/2024 (Previous: 07/08/2022)
<1> ; -----
<1> ; Beginning: 24/01/2016
<1> ; -----
<1> ; Assembler: NASM version 2.15 (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)

```

```

24      <1> ; <Bell Laboratories (17/3/1972)>
25      <1> ; <Preliminary Release of UNIX Implementation Document>
26      <1> ; (Section E10 (17/3/1972) - ux.s)
27      <1> ; *****
28      <1> ; Ref: Retro UNIX 386 v1.2 Kernel (v0.2.2.3) - ux.s - 15/07/2022
29      <1>
30      <1> alignb 2
31      <1>
32      <1> %if 0
33      <1>
34      <1> inode:
35      <1>      ; 11/03/2013.
36      <1>      ; Derived from UNIX v1 source code 'inode' structure (ux).
37      <1>      ; i.
38      <1>      ;
39      <1>      ; i.flgs: resw 1
40      <1>      ; i.nlks: resb 1
41      <1>      ; i.uid: resb 1
42      <1>      ; i.size: resw 1 ; size
43      <1>      ; i.dskp: resw 8 ; 16 bytes
44      <1>      ; i.ctim: resd 1
45      <1>      ; i.mtim: resd 1
46      <1>      ; i.rsvd: resw 1 ; Reserved (ZERO/Undefined word for UNIX v1)
47      <1>
48      <1>      ; 26/01/2020
49      <1>      ; Retro UNIX 386 v2.0 - Modified UNIX v7 inode model
50      <1>      ; (15/09/2029 .. 18/12/2019)
51      <1>
52      <1>      i.flgs: resw 1 ; /* mode and type of file */
53      <1>      i.nlks: resw 1 ; /* number of links to file */
54      <1>      i.uid: resw 1 ; /* owner's user id */ - 0 to 65535 -
55      <1>      i.gid: resb 1 ; /* owner's group id */ - 0 to 255 -
56      <1>      i.size_h: resb 1 ; /* number of bytes in file */ ; byte 5
57      <1>      i.size: resd 1 ; size ; /* number of bytes in file */
58      <1>      i.dskp: resd 10 ; 40 bytes ; /* disk block addresses */
59      <1>      i.atim: resd 1 ; /* time last accessed */
60      <1>      i.mtim: resd 1 ; /* time last modified */
61      <1>      i.ctim: resd 1 ; /* time created */
62      <1>
63      <1> I_SIZE equ $ - inode
64      <1>
65      <1> %endif
66      <1>
67      <1> process:
68      <1>      ; 23/07/2022 - TRDOS 386 kernel v2.0.5
69      <1>      ; 27/02/2022
70      <1>      ; 12/01/2022 - Retro UNIX 386 v1.2
71      <1>      ; 19/12/2016
72      <1>      ; 21/05/2016
73      <1>      ; 19/05/2016 - TRDOS 386 (TRDOS v2.0)
74      <1>      ; 06/05/2015 - Retro UNIX 386 v1
75      <1>      ; 11/03/2013 - 05/02/2014 (Retro UNIX 8086 v1)
76      <1>      ; Derived from UNIX v1 source code 'proc' structure (ux).
77      <1>      ; p.
78      <1>
79      <1>      p.pid: resw nproc
80      <1>      p.ppid: resw nproc
81      <1>      ; p.break: resw nproc ; 12/01/2022 (p.break is not used)
82      <1>      p.ttyc: resb nproc ; console tty in Retro UNIX 8086 v1.
83      <1>      ; 27/02/2022 (p.waitc is not used)
84      <1>      ; p.waitc: resb nproc ; waiting channel in Retro UNIX 8086 v1.
85      <1>      p.link: resb nproc
86      <1>      p.stat: resb nproc
87      <1>
88      <1>      ; 06/05/2015 (Retro UNIX 386 v1 feature only !)
89      <1>      p.upage: resd nproc ; Physical address of the process's
90      <1>      ; 'user' structure
91      <1>      ; 21/05/2016
92      <1>      ; 19/05/2016 (TRDOS 386 feature only!)
93      <1>      p.timer: resb nproc ; number of timer events of the processs
94      <1>
95      <1>      ; 19/12/2016
96      <1>      p.tcb: resd nproc ; timer callback service address (if > 0)
97      <1>
98      <1>      ; 18/08/2024 (TRDOS 386 v2.0.9)
99      <1>      p.exitt: resb nproc ; exit code of the child
100     <1>
101     <1> P_SIZE equ $ - process
102     <1>
103     <1> ; fsp table (original UNIX v1)
104     <1>
105     <1> ; Entry
106     <1>      15 0
107     <1>      1 |-----|
108     <1>      | r/w | i-number of open file |
109     <1>      |-----|
110     <1>      | device number |
111     <1>      |-----|
112     <1>      (*) | offset pointer, i.e., r/w pointer to file |
113     <1>      |-----|
114     <1>      | flag that says | number of processes |
115     <1>      | file deleted | that have file open |
116     <1>      |-----|
117     <1>      2 |-----|
118     <1>      |-----|
119     <1>      |-----|
120     <1>      |-----|
121     <1>      |-----|
122     <1>      |-----|
123     <1>      |-----|
124     <1>      3 |-----|
125     <1>      |-----|
126     <1>      |-----|
127     <1>
128     <1> ; (*) Retro UNIX 386 v1 modification: 32 bit offset pointer
129     <1>
130     <1> ; 27/03/2020 - Retro UNIX 386 v2 - FSP (OPEN FILES) TABLE
131     <1>
132     <1> ; Entry
133     <1>      15 7 0
134     <1>      1 |-----| |
135     <1>      | i-number of open file |
136     <1>      |-----|
137     <1>      | high word of 32 bit i-number |
138     <1>      |-----|
139     <1>      | open mode & status | device number |
140     <1>      |-----|
141     <1>      | reserved byte | open count |
142     <1>      |-----|
143     <1>      | offset pointer, i.e., r/w pointer to file |
144     <1>      |-----|
145     <1>      | 64 bit file offset pointer (bit 16-31) |
146     <1>      |-----|
147     <1>      | 64 bit file offset pointer (bit 32-47) |

```

```

148 <1> ;
149 <1> ;
150 <1> ; 64 bit file offset pointer (bit 48-63)
151 <1> ; 2
152 <1> ;
153 <1> ;
154 <1> ;
155 <1> ;
156 <1> ;
157 <1> ;
158 <1> ;
159 <1> ;
160 <1> ;
161 <1> %if 0
162 <1>
163 <1> ; (Retro UNIX 386 v1.2 - ux.s - 15/07/2022)
164 <1> ; 22/11/2021
165 <1> ; 21/07/2021 - Retro UNIX 386 v2 open file structure revision
166 <1>
167 <1> struct file ; open files (fsp) structure ; (*)
168 <1> .inode: resw 1 ; inode number of open file (32 bit)
169 <1> .i32: resw 1 ; higher word of inode number (reserved)
170 <1> .drive: resb 1 ; logical drive (disk) number
171 <1> .flags: resb 1 ; open mode and status
172 <1> .count: resb 1 ; number of processes that have file open
173 <1> .rsvd: resb 1 ; reserved byte (for next versions)
174 <1> .mnt: resb 1 ; mnttab index+1 (0 = not mounted)
175 <1> .offset: resd 1 ; file offset/pointer (64 bit)
176 <1> .o64: resd 1 ; higher 32 bit of file offset
177 <1> .size: ; = 16
178 <1> endstruc
179 <1>
180 <1> %endif
181 <1>
182 <1> ; 23/07/2022
183 <1> ; fsp: resb nfiles * 16 ; (*)
184 <1>
185 <1> idev: resb 1
186 <1> cdev: resb 1
187 <1>
188 <1> ; 15/04/2015
189 <1> ; fsp: resb nfiles * 10 ; 11/05/2015 (8 -> 10)
190 <1> ; idev: resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
191 <1> ; cdev: resw 1 ; device number is 1 byte in Retro UNIX 8086 v1 !
192 <1>
193 <1> ; 18/05/2015
194 <1> ; 26/04/2013 device/drive parameters (Retro UNIX 8086 v1 feature only!)
195 <1> ; 'UNIX' device numbers (as in 'cdev' and 'u.cdrv')
196 <1> ; 0 -> root device (which has Retro UNIX 8086 v1 file system)
197 <1> ; 1 -> mounted device (which has Retro UNIX 8086 v1 file system)
198 <1> ; 'Retro UNIX 8086 v1' device numbers: (for disk I/O procedures)
199 <1> ; 0 -> fd0 (physical drive, floppy disk 1), physical drive number = 0
200 <1> ; 1 -> fd1 (physical drive, floppy disk 2), physical drive number = 1
201 <1> ; 2 -> hd0 (physical drive, hard disk 1), physical drive number = 80h
202 <1> ; 3 -> hd1 (physical drive, hard disk 2), physical drive number = 81h
203 <1> ; 4 -> hd2 (physical drive, hard disk 3), physical drive number = 82h
204 <1> ; 5 -> hd3 (physical drive, hard disk 4), physical drive number = 83h
205 <1>
206 <1> rdev: resb 1 ; root device number ; Retro UNIX 8086 v1 feature only!
207 <1> ; as above, for physical drives numbers in following table
208 <1> mdev: resb 1 ; mounted device number ; Retro UNIX 8086 v1 feature only!
209 <1>
210 <1> ; 23/07/2022
211 <1> ; ; 15/04/2015
212 <1> ; active: resb 1
213 <1> ; resb 1 ; 09/06/2015
214 <1>
215 <1> ; 23/07/2022
216 <1> ; mnti: resw 1
217 <1> ; ; 07/08/2022
218 <1> ; mpid: resw 1
219 <1> ; rootdir: resw 1
220 <1> ; rootdir: resd 1
221 <1>
222 <1> ; 21/05/2016 - TRDOS 386 (TRDOS v2.0) - priority levels, 3 run queues
223 <1> runq:
224 <1> runq_event: resw 1 ; high priority, 'run for event' ; 2
225 <1> runq_normal: resw 1 ; normal/regular priority, 'run as regular' ; 1
226 <1> runq_background: resw 1 ; low priority, 'run on background' ; 0
227 <1>
228 <1> ; 23/07/2022
229 <1> ; imod: resb 1
230 <1> ; smod: resb 1
231 <1> ; mmmod: resb 1
232 <1> ; sysflg: resb 1
233 <1> ; resb 1
234 <1>
235 <1> alignb 4
236 <1>
237 <1> user:
238 <1> ; 23/07/2022 - TRDOS 386 Kernel v2.0.5
239 <1> ; 04/12/2021 - Retro UNIX 386 v1.2
240 <1> ; 13/01/2017
241 <1> ; 19/12/2016
242 <1> ; 21/05/2016 - TRDOS 386 (TRDOS v2.0)
243 <1> ; [u.pri] usage method modification
244 <1> ; 04/12/2015
245 <1> ; 18/10/2015
246 <1> ; 12/10/2015
247 <1> ; 21/09/2015
248 <1> ; 24/07/2015
249 <1> ; 16/06/2015
250 <1> ; 09/06/2015
251 <1> ; 11/05/2015
252 <1> ; 16/04/2015 (Retro UNIX 386 v1 - 32 bit modifications)
253 <1> ; 10/10/2013
254 <1> ; 11/03/2013.
255 <1> ; Derived from UNIX v1 source code 'user' structure (ux).
256 <1> ; u.
257 <1>
258 <1> u.sp: resd 1 ; esp (kernel stack at the beginning of 'sysent')
259 <1> u.usp: resd 1 ; esp (kernel stack points to user's registers)
260 <1> u.r0: resd 1 ; eax
261 <1> u.cdir: resw 1
262 <1> u.cdrv: resw 1 ; 23/07/2022
263 <1> u.cdrv: resb 1 ; 23/07/2022
264 <1> resb 1
265 <1> u.fp: resb 10
266 <1> ; u.fp: resb OPENFILES ; 23/07/202
267 <1> ; u.fsp: ; 23/07/2022
268 <1> u.fofp: resd 1
269 <1> u.dirp: resd 1
270 <1> u.namep: resd 1
271 <1> u.off: resd 1

```

```

272      <1>      ;      resd 1 ; 23/07/2022 (64 bit fptr)
273 00018E78 ???????? <1>      u.base: resd 1
274 00018E7C ???????? <1>      u.count: resd 1
275 00018E80 ???????? <1>      u.nread: resd 1
276 00018E84 ???????? <1>      u.break: resd 1 ; break
277      <1>      ; 10/01/2017 (TRDOS 386, relocation and dword alignment)
278      <1>      ; tty number (rtty, rcvt, wtty)
279 00018E88 ??? <1>      u.ttyp: resw 1
280 00018E8A ?? <1>      u.ttyn: resb 1 ; 28/07/2013 - Retro Unix 8086 v1 feature only !
281 00018E8B ?? <1>      u.mode: resb 1 ; 23/07/2022
282      <1>      ;u.resb: resb 1 ; 10/01/2017 (TRDOS 386, temporary)
283 00018E8C <res 10h> <1>      u.dirbuf: resb 16 ; 04/12/2015 (10 -> 16)
284      <1>      ;u.pri: resw 1 ; 14/02/2014
285 00018E9C ?? <1>      u.quant: resb 1 ; Retro UNIX 8086 v1 Feature only ! (uquant)
286 00018E9D ?? <1>      resb 1 ; 23/07/2022
287 00018E9E ?? <1>      u.pri: resb 1 ; Modification: 21/05/2016 (priority levels: 0, 1, 2)
288 00018E9F ?? <1>      resb 1 ; 23/07/2022
289 00018EA0 ??? <1>      u.intr: resw 1
290 00018EA2 ??? <1>      u.quit: resw 1
291      <1>      ;u.emt: resw 1 ; 10/10/2013
292      <1>      ;u.ilgins: resw 1 ; 10/01/2017
293      <1>      ;u.cdrv: resw 1 ; cdev
294 00018EA4 ?? <1>      u.bsys: resb 1
295 00018EA5 ?? <1>      u.uno: resb 1
296      <1>      ; 23/07/2022
297      <1>      ;u.uid: resw 1 ; uid ; 27/03/2021 - Retro UNIX 386 v2
298      <1>      ;u.ruid: resw 1 ; 16 bit uid
299      <1>      ;u.gid: resb 1 ; gid ; 27/03/2021 - Retro UNIX 386 v2
300      <1>      ;u.rgid: resb 1
301      <1>      ; 23/07/2022
302      <1>      ;u.procp: resd 1 ; /* pointer to proc structure */
303 00018EA6 ?? <1>      u.uid: resb 1 ; uid
304 00018EA7 ?? <1>      u.ruid: resb 1
305 00018EA8 ???????? <1>      u.upage: resd 1 ; 16/04/2015 - Retro Unix 386 v1 feature only !
306 00018EAC ???????? <1>      u.pgdir: resd 1 ; 09/03/2015 (page dir addr of process)
307 00018EB0 ???????? <1>      u.ppgdir: resd 1 ; 06/05/2015 (page dir addr of the parent process)
308 00018EB4 ???????? <1>      u.pbase: resd 1 ; 20/05/2015 (physical base/transfer address)
309 00018EB8 ??? <1>      u.pcount: resw 1 ; 20/05/2015 (byte -transfer- count for page)
310      <1>      ;u.pncount: resw 1
311      <1>      ; 16/06/2015 (byte -transfer- count for page, 'namei', 'mkdir')
312      <1>      ;u.pnbase: resd 1
313      <1>      ; 16/06/2015 (physical base/transfer address, 'namei', 'mkdir')
314      <1>      ; 09/06/2015
315 00018EBA ?? <1>      u.kcall: resb 1 ; The caller is 'namei' (dskr) or 'mkdir' (dskw) sign
316 00018EBB ?? <1>      u.brwdev: resb 1 ; Block device number for direct I/O (bread & bwrite)
317      <1>      ; 24/07/2015 - 24/06/2015
318      <1>      ;u.args: resd 1 ; arguments list (line) offset from start of [u.upage]
319      <1>      ; (arg list/line is from offset [u.args] to 4096 in [u.upage])
320      <1>      ; ([u.args] points to argument count -argc- address offset)
321      <1>      ; 24/06/2015
322      <1>      ;u.core: resd 1 ; physical start address of user's memory space (for sys exec)
323      <1>      ;u.ecore: resd 1 ; physical end address of user's memory space (for sys exec)
324      <1>      ;;
325      <1>      ; 19/08/2024 ; TRDOS 386 v2.0.9 ; STDIN/STDOUT/STDERR
326 00018EBC ?? <1>      u.stdin: resb 1 ; 0 is tty-r or > 0 is file (redirection)
327 00018EBD ?? <1>      u.stdout: resb 1 ; 0 is tty-w or > 0 is file (redirection)
328 00018EBE ?? <1>      u.ungetc: resb 1 ; u.getc is valid if u.ungetc > 0
329 00018EBF ?? <1>      u.getc: resb 1 ; last char read on stdin
330      <1>      ;;
331      <1>      ; last error number
332 00018EC0 ???????? <1>      u.error: resd 1 ; 28/07/2013 - 09/03/2015
333      <1>      ; Retro UNIX 8086/386 v1 feature only!
334      <1>      ; 21/09/2015 (debugging - page fault analyze)
335 00018EC4 ???????? <1>      u.pfcount: resd 1 ; page fault count for (this) process (for sys geterr)
336      <1>      ; 19/12/2016 (TRDOS 386)
337 00018EC8 ???????? <1>      u.tcb: resd 1 ; Timer callback address/flag which will be used by timer int
338      <1>      ; 13/01/2017 (TRDOS 386)
339 00018ECC ?? <1>      u.t_lock: resb 1 ; Timer interrupt (callback) lock (unlocked by 'sysrele')
340 00018ECD ?? <1>      u.t_mode: resb 1 ; running mode during timer interrupt (0= system, 0FFh= user)
341      <1>      ; 26/02/2017 (TRDOS 386)
342 00018ECE ?? <1>      u.irqc: resb 1 ; Count of IRQ callback services (IRQs in use)
343      <1>      ; 28/02/2017 (TRDOS 386)
344 00018ECF ?? <1>      u.irqwait: resb 1 ; IRQ waiting for callback service flag (IRQ number, If > 0)
345 00018ED0 ?? <1>      u.r_lock: resb 1 ; 'IRQ callback service is in progress' flag (IRQ lock)
346 00018ED1 ?? <1>      u.r_mode: resb 1 ; running mode during hardware interrupt
347      <1>      ; 23/07/2022
348 00018ED2 ?? <1>      u.exit: resb 1 ; exit code
349      <1>      ; 27/02/2017 (TRDOS 386)
350 00018ED3 ?? <1>      u.fpsave: resb 1 ; TRDOS 386, 'save/restore FPU registers' flag
351      <1>      alignb 4
352      <1>      ; !! wrong sizing in TRDOS 386 v2.0.4 (in 'ubss.s', 28/02/2017) !!
353      <1>      ;u.fpregs: resb 94 ; 94 byte area for saving and restoring FPU registers
354      <1>      ; 23/07/2022
355 00018ED4 <res 6Ch> <1>      u.fpregs: resb 108 ; 108 byte area for saving and restoring FPU registers
356      <1>
357      <1>      alignb 4
358      <1>
359      <1>      U_SIZE equ $ - user
360      <1>
361      <1>      ; 18/10/2015 - Retro UNIX 386 v1 (local variables for 'namei' and 'sysexec')
362 00018F40 ???????? <1>      pcore: resd 1 ; physical start address of user's memory space (for sys exec)
363 00018F44 ???????? <1>      ecore: resd 1 ; physical address of user's stack/last page (for sys exec)
364 00018F48 ???????? <1>      nbase: resd 1 ; physical base address for 'namei' & 'sysexec'
365      <1>      ; 23/07/202 - TRDOS 386 Kernel v2.0.5
366      <1>      ;ncount: resw 1
367 00018F4C ???????? <1>      ncount: resd 1 ; remain byte count in page for 'namei' & 'sysexec'
368      <1>      ;argc: resw 1
369 00018F50 ???????? <1>      argc: resd 1 ; argument count for 'sysexec'
370 00018F54 ???????? <1>      argv: resd 1 ; argument list (recent) address for 'sysexec'
371      <1>
372      <1>      ; 03/06/2015 - Retro UNIX 386 v1 Beginning
373      <1>      ; 07/04/2013 - 31/07/2013 - Retro UNIX 8086 v1
374 00018F58 ?? <1>      rw: resb 1 ; Read/Write sign (iget)
375      <1>      ; 23/07/2022
376 00018F59 ?????? <1>      resb 3
377      <1>
378      <1>      alignb 4
379      <1>
380      <1>      ; 24/04/2016
381 00018F5C ???????? <1>      ii: resd 1 ; first cluster of the program file
382 00018F60 ???????? <1>      i.size: resd 1 ; size of the program file
4009
4010      alignb 4
4011
4012      ; 23/05/2016 (TRDOS 386)
4013      ; 14/10/2015 (Retro UNIX 386 v1, 'unix386.s')
4014 00018F64 ???????? cr3reg: resd 1 ; cr3 register content at the beginning of the timer
4015      ; (or RTC) interrupt handler.
4016
4017      ; 10/12/2016 (callback)
4018      ; 10/06/2016
4019      ; 19/05/2016
4020      ; 18/05/2016 - TRDOS 386 feature only !
4021 00018F68 <res 100h> timer_set: resd 16*4 ; 256 bytes memory space for 16 timer events

```



```

4022 ; Timer Event Structure: (max. 16 timer events, 16*16 bytes)
4023 ; Owner: resb 1 ; 0 = free
4024 ; ;>0 = process number (u.uno)
4025 ; Callback: resb 1 ; 0 = response byte address (phy)
4026 ; ; 1 = callback address (virtual)
4027 ; Interrupt: resb 1 ; 0 = Timer interrupt (or none)
4028 ; ; 1 = Real Time Clock interrupt
4029 ; Response: resb 1 ; 0 to 255, signal return value
4030 ; Count Limit: resd 1 ; count of ticks (total/set)
4031 ; Current Count: resd 1 ; count of ticks (current)
4032 ; Response Addr: resd 1 ; response byte (pointer) address
4033 ; ; (or callback -user service- address)
4034 ;
4035 ;
4036 ; 17/04/2021
4037 ; (memory page swap parameters are disabled as temporary)
4038 ;
4039 ; Memory (swap) Data (11/03/2015)
4040 ; 09/03/2015
4041 ; swpq_count: resw 1 ; count of pages on the swap queue
4042 ; swp_drv: resd 1 ; logical drive description table address of the swap drive/disk
4043 ; swpd_size: resd 1 ; size of swap drive/disk (volume) in sectors (512 bytes).
4044 ; swpd_free: resd 1 ; free page blocks (4096 bytes) on swap disk/drive (logical)
4045 ; swpd_next: resd 1 ; next free page block
4046 ; swpd_last: resd 1 ; last swap page block
4047
4048 alignb 4
4049
4050 ; 10/07/2015
4051 ; 28/08/2014
4052 00019068 ???????? error_code: resd 1
4053 ; 29/08/2014
4054 0001906C ???????? FaultOffset: resd 1
4055 ; 21/09/2015
4056 00019070 ???????? PF_Count: resd 1 ; total page fault count
4057 ; ; (for debugging - page fault analyze)
4058 ; ; 'page_fault_handler' (memory.inc)
4059 ; ; 'sysgeterr' (u9.s)
4060
4061 ; 29/04/2016 (TRDOS 386 = TRDOS v2.0)
4062 ; 22/08/2015 (Retro UNIX 386 v1)
4063 buffer:
4064 00019074 ?????????????? resb 8
4065 readi_buffer:
4066 0001907C <res 200h> resb 512
4067 0001927C ?????????????? resb 8
4068 writei_buffer:
4069 00019284 <res 200h> resb 512
4070 ; 24/10/2016
4071 00019484 ?????????????? resb 8
4072 rw_buffer:
4073 0001948C <res 800h> resb 2048 ; general purposed, r/w sector buffer
4074
4075 %if 1
4076 ; 17/01/2021
4077 00019C8C <res 80h> edid_info: resb 128 ; VESA EDID (monitor capabilities) info
4078 ; 28/11/2020
4079 00019D0C ?? pmi32: resb 1 ; (>0) use VESA VBE3 protected mode calls
4080 00019D0D ?? vbe_mode_x: resb 1 ; VESA VBE3 video bios mode set options
4081 00019D0E ??? video_mode: resw 1 ; VESA VBE3 video mode (with option flags)
4082 ; 30/11/2020
4083 00019D10 ???????? vbe3bios_addr: resd 1 ; new (writable mem) address of VBE3 bios
4084 ; 02/12/2020
4085 00019D14 ???????? pmid_addr: resd 1 ; PMInfoBlock ('PMID') linear address
4086 ; 14/01/2021
4087 ; 06/12/2020 ; VESA VBE 3 video state
4088 ; vbe3stbufsize: resw 1 ; video regs/dac/bios state buffer size
4089 ; ; block size in bytes
4090 ; 16/01/2021
4091 00019D18 ??? vbe3stbsflags: resw 1 ; video regs/dac/bios state buffer size
4092 ; ; pointer flags for buffer state options
4093 %endif
4094
4095 %if 1
4096 ; 10/12/2020
4097 LFB_Info:
4098 00019D1A <res 10h> resb 16 ; Linear Frame Buffer info block
4099
4100 ; 24/11/2020 - TRDOS 386 v2.0.3
4101 ; BOCHS/PLEX86 VESA VBE3 MODE INFO extension to TRDOS 386 v2 kernel
4102 MODE_INFO_LIST:
4103 00019D2A <res 44h> resb 68 ; mode + 66 byte VESA vbe3 mode info (4F01h)
4104 %endif
4105
4106 ; 05/01/2021
4107 00019D6E ?? ufont: resb 1 ; (VGA graphics) user font flags
4108 ; ; bit 7 - permission flag for int 31h
4109 ; ; bit 4 - 8x16 user font ready/loaded flag
4110 ; ; bit 3 - 8x8 user font ready/loaded flag
4111 ; ; bit 1 - select 8x16 user font (sysvideo)
4112 ; ; bit 0 - select 8x8 user font (sysvideo)
4113 00019D6F ?? resb 1 ; 19/01/2021
4114 ; 17/01/2021
4115 00019D70 ?? srvsf: resb 1 ; 'save restore video state' permission flag
4116 ; 18/01/2021
4117 00019D71 ?? srvs0: resb 1 ; video state buffer save/restore option
4118 00019D72 ???????? VideoStateID: resd 1 ; used to verify state saved by same prog
4119 ; 29/01/2021
4120 00019D76 ??? v_width: resw 1 ; screen (display page) width
4121 00019D78 ?? v_ops: resb 1 ; 'sysvideo' graphics data transfer option
4122 00019D79 ?? v_bpp: resb 1 ; bits per pixels ('sysvideo')
4123 00019D7A ???????? v_mem: resd 1 ; video memory ('sysvideo')
4124 00019D7E ???????? v_siz: resd 1 ; video page size ('sysvideo')
4125 00019D82 ???????? v_str: resd 1 ; window start adress ('sysvideo')
4126 00019D86 ???????? v_end: resd 1 ; window end (end+1) adress ('sysvideo')
4127 ; 31/01/2021
4128 ; 01/01/2021
4129 ; maskbuff: ; resd 1 ; user's bitmask buffer addr ('sysvideo')
4130 00019D8A ???????? maskcolor: resd 1 ; VGA/SVGA pixel mask color ('sysvideo')
4131 ; 27/02/2021
4132 00019D8E ???????? pixcount: resd 1 ; pixel count ('sysvideo' window ops)
4133 ; 02/02/2021
4134 00019D92 ?????????????? buffer8: resd 2 ; 8 bytes small buffer for 'sysvideo'
4135
4136 bss_end:
4137
4138 ; 27/12/2013
4139 _end: ; end of kernel code

```